HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

DIPLOMARBEIT

Herr
Dominik Baumann

Laufzeitsystem für Software-Objekte auf Basis von IEC 61131

2016

Fakultät **Elektro- und Informationstechnik**

DIPLOMARBEIT

Laufzeitsystem für Software-Objekte auf Basis von IEC 61131

Autor:
Dominik Baumann

Studiengang:
Technische Informatik

Seminargruppe:
KT11wIA-F

Erstprüfer:
Prof. Dr.-Ing. Swen Schmeißer

Zweitprüfer:
Dipl.-Ing.(FH) Martin Klotz

Mittweida, Jänner 2016

Bibliografische Angaben

Baumann, Dominik: Laufzeitsystem für Software-Objekte auf Basis von IEC 61131, 53 Seiten, 18 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Elektro- und Informationstechnik

Diplomarbeit, 2016

Referat

In der vorliegenden Arbeit wird die Konzeption, Entwicklung und Integration eines flexiblen Laufzeitsystemes für Speicherprogrammierbare-Steuerungen vorgestellt. Hauptziel ist ein herstellerunabhängiges System zu schaffen, welches von den Systemtechnikern flexibel parametriert werden kann. Es sind keine Anpassungen im SPS-Programm zur Änderung der Funktionalitäten notwendig. Die Funktionalitäten werden durch Software-Objekte auf den Steuerungen repräsentiert. Diese können aus dem Leitsystem geladen und vollständig konfiguriert werden. Das Laufzeitsystem wird mit einem IEC 61131 basierenden Entwicklungswerkzeug des jeweiligen Steuerungsherstellers erstellt. In der vorliegenden Implementierung wurde Beckhoff Twincat2 für die Steuerungsebene und Siemens WinCC OA als Leitsystem verwendet. Die Software-Objekte können beliebige Funktionalitäten abarbeiten und in begrenztem Maße miteinander verknüpft werden. Außerdem können die Objekte wahlweise in der Steuerung oder im Leitsystem ablaufen.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Vorwort

Durch den rasant wachsenden Markt an industriellen Steuerungssystemen, wird es für Techniker zunehmend schwieriger alle notwendigen Werkzeuge zu beherrschen. In modernen Industrieanlagen werden verschiedenste Systeme unterschiedlicher Hersteller gemeinsam eingesetzt. Die Techniker sollen dabei jedoch stets für einen reibungslosen Betrieb sorgen und alle Probleme in kurzer Zeit beheben. Außerdem sind die Anlagen meist in einem größeren Verbund verschalten und kommunizieren mit übergeordneten Systemen.

Wäre es nicht schön, verschiedene Steuerungssysteme mit einem Werkzeug einfach und sicher parametrieren zu können?

... so könnten getestete Funktionen, ohne Zutun von spezialisierten Steuerungsentwicklern, von den Technikern selbst auf den Steuerungen in Betrieb genommen und gewartet werden. Weniger Fehler, schnellere Problembehebung und motiviertere Mitarbeiter wären die Folge.

„Durch wieviel Kompliziertheit muss man sich durchringen, bis man endlich zur Einfachheit gelangt.“

Marie von Ebner-Eschenbach

1 Einleitung

1.1 Motivation

In meiner langjährigen Tätigkeit als Leitsystemtechniker bei der Firma Plansee in Reutte habe ich mich ausgiebig mit den Themen der Datenerfassung und Anlagensteuerung beschäftigt. Dabei wurde schnell klar, dass ein kosteneffizienter Betrieb eines solchen Systems nur mit standardisierten Hard- und Softwarekomponenten möglich ist. Um weitgehend herstellerunabhängig zu sein, wollten wir nicht die Komponenten selbst, sondern die Prozesse standardisieren. Dazu muss eine flexible und skalierbare Softwareplattform geschaffen werden. Durch die vielschichtigen Aufgaben in diesem Bereich brauchen wir Komponenten die von den Nutzern wie ein Baukasten zusammengesteckt werden können. Dabei muss die Konfiguration und Parametereingabe möglichst einfach und weitgehend automatisch funktionieren. Außerdem müssen in der Feldebene, verschiedenste Systeme eingebunden werden um den Bestand an Anlagen abdecken zu können. Da am Markt kein derartiges Gesamtsystem eingekauft werden konnte, mussten die notwendigen Funktionalitäten auf Basis einer Leittechnik-Plattform selbst entwickelt werden.

Im Rahmen des Praxisprojekt II habe ich bereits die „verteilte Feldebene für das Plansee Meldesystem“ ausführlich beschrieben. Hierbei geht es um die einfache und systematische Konfiguration und Abfrage der Eingangs- und Ausgangs-Hardware. Das Feldbussystem kann so einfach parametriert und die Signale im Leitsystem verwendet werden. [23]

Allerdings ist mit der verteilten Feldebene keine weiterführende Datenverarbeitung im Feld möglich, da benutzerdefinierter Code nur im Leitsystem ablaufen kann. Für viele Aufgaben in der Steuerungstechnik ist die Datenverarbeitung in den lokalen Laufzeitsystemen der Feldebene allerdings unumgänglich. Um diesen Nachteil der *„dummen“* Feldebene zu beheben, soll das Plansee-Meldesysteme um ein Laufzeitsystem für Software-Objekte ergänzt werden.

1.2 Vorstellung der Plansee Gruppe

Seit über 90 Jahren entwickelt, produziert und vermarktet die Plansee Gruppe Produkte basierend auf den Hochtechnologie-Werkstoffen Molybdän und Wolfram. Wolfram, Tantal und Molybdän sind neben Kohlenstoff die hitzebeständigsten Werkstoffe. Mit ihrem hohen Schmelzpunkt eignen sie sich als Heizer in Hochtemperaturöfen oder als Schmelztiegel für die Saphir Herstellung. In elektrischen Schaltkontakten punkten sie mit sehr guter elektrischer und thermischer Leitfähigkeit. Und sie überzeugen als elek-

trisch leitende Schicht in Flachbildschirmen und Solarzellen. Zudem haben sie viele weitere gefragte Eigenschaften wie hohe Dichte und Reinheit, hohe Verschleiß- und Korrosionsbeständigkeit sowie die Fähigkeit, Strahlen zu absorbieren. Die Plansee Gruppe hat im Geschäftsjahr 2013/14 einen Jahresumsatz von 1,2 Milliarden Euro erwirtschaftet. Von den gesamt 6060 Mitarbeitern sind am Hauptstandort in Reutte derzeit ca. 2300 Mitarbeiter beschäftigt. [35]

1.3 Geschichte des Plansee-Meldesystems

Im Jahre 2003 wurde am Standort Reutte das Plansee-Meldesystem in Betrieb genommen. Dieses System sollte eine einfache und flexible Möglichkeit zur Erfassung von Messwerten und Signalen im gesamten Werksgelände ermöglichen. Es bestand aus vier speicherprogrammierbaren Steuerungen (sogenannte Knoten) mit Ethernet Anschaltung. Alle relevanten Messwerte am Standort Reutte wurden auf diese Knoten aufgeschaltet und zu einem zentralen Leitsystem übertragen. Dazu musste für jedes Signal das SPS-Programm angepasst und die Anzeige in der Leitsystem-Software implementiert werden. Mit den Jahren ist das System stark angewachsen und umfasst mittlerweile über hundert Knoten. Zusätzlich wurden weitere Subsysteme, wie Videoüberwachung, Zugangskontrolle und Brandmeldeanlagen, angeschlossen. Im Leitsystem sind nun mehr als 10.000 Datenpunkte visualisiert. Dabei sind die Knoten über verschiedenste Kommunikationswege, wie Richtfunk, DSL [1], Glasfaser und Ethernet, angebunden. Um die Überwachung kritischer Anlagen sicherzustellen, muss das System ständig verfügbar sein. Außerdem sind gesetzliche Auflagen und Normen in diesem Bereich einzuhalten.

Mit der Zeit wurde klar, dass die steigende Komplexität und Menge der zu übertragenden Daten mit dem bestehenden System nicht mehr bewältigt werden können. Das System ist für einen effizienten Betrieb zu kompliziert aufgebaut. Es muss eine bessere Methode zur Kopplung zwischen Feldebene und Leitsystem gefunden werden. Aus dem Störmeldesystem ist ein zentrales Datenerfassungswerkzeug geworden. Von Abrechnung bis Zählerverwaltung, liefert das System Daten für eine Vielzahl von Systemen und Prozessen. Um auch in Zukunft diese Aufgaben erfüllen zu können, ist eine Anpassung der derzeitigen Struktur notwendig.

1.3.1 Das bestehende System

Derzeit wird als Leitsystem die Software SIMATIC WinCC Open Architecture (WinCC OA) der österreichischen Firma ETM verwendet. Die Firma ETM professional control GmbH, mit Sitz in Eisenstadt, ist Teil der Siemens Gruppe. WinCC OA bietet eine flexible und hoch skalierbare Plattform zur Prozessvisualisierung und -Steuerung.

[1] Digital Subscriber Line

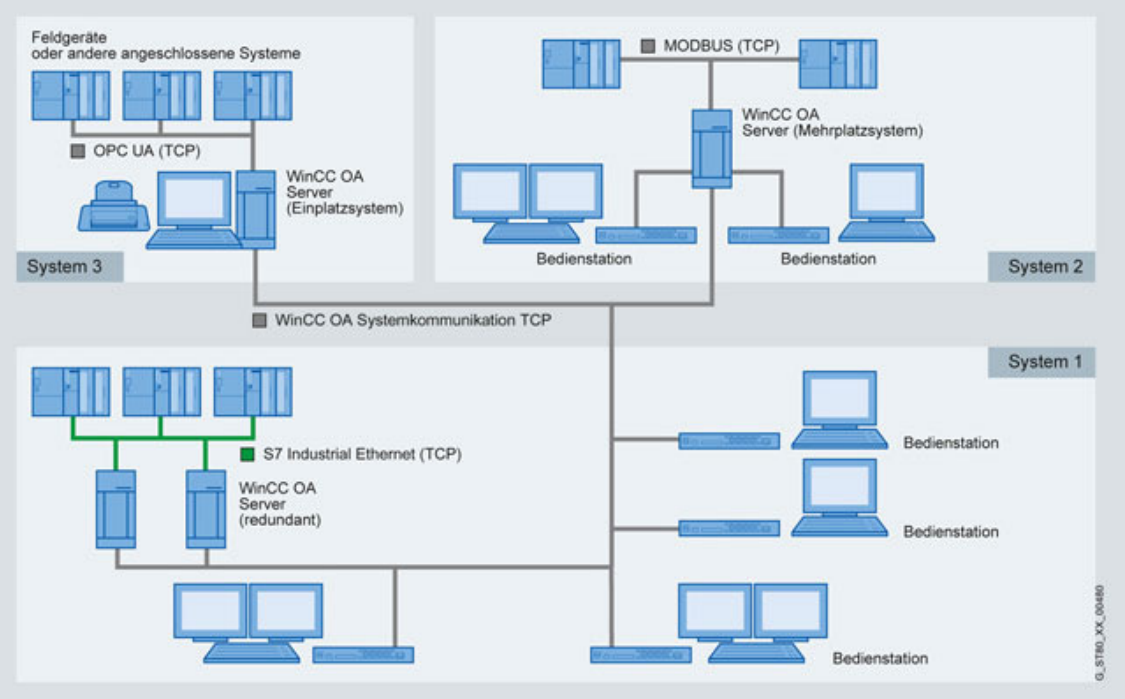

Abbildung 1.1: Schema WinCC OA Schema mit drei Systemen [24]

Anpassungen und spezielle Erweiterungen können über die integrierte Scriptsprache CONTROL oder direkt in C++ [2] geschrieben werden. Außerdem bietet das System leistungsfähige Grafikfunktionen auf Basis von Qt [3]. Für die Kommunikation mit der Feldebene stehen eine Vielzahl von Treibern zur Verfügung. Derzeit werden Modbus-TCP, IEC 60870-5-104 und S7-Kommunikation genutzt. Alle Programmkomponenten, die sogenannten Manager, kommunizieren über Netzwerkverbindungen mit einem zentralen Event-Manager. Abbildung 1.1 zeigt ein allgemeines Schema eines verteilten WinCC OA Systems bestehend aus drei Einzelsystemen. Das bestehende Plansee-Meldesystem am Standort Reutte umfasst derzeit drei Systeme. Für die Erfassung der Signale und Meldungen im Feld werden sowohl Beckhoff- als auch Siemens-Steuerungen eingesetzt. Die Siemens-Steuerungen verfügen, wie die Steuerungen von Beckhoff, über eine Netzwerkanschaltung und kommunizieren über den WinCC OA-S7 Treiber mit dem Leitsystem.

Der Großteil aller Signale wird mit Beckhoff-Steuerungen erfasst. Diese bieten ein flexibles Feldbussystem mit einer Vielzahl von Ein- und Ausgangsbaugruppen. Außerdem werden alle gängigen Kommunikationsprotokolle unterstützt. Außerdem gibt es Steuerungen in verschiedenen Leistungsklassen. In der Abbildung 1.2 ist der Aufbau eines Beckhoff-Klemmbus-Systems dargestellt. Die elektrischen Signale werden an den passenden Klemmen angeschlossen und von der Steuerung verarbeitet. Anschließend wer-

[2] C++ ist eine Programmiersprache, die als Erweiterung der Programmiersprache C entwickelt wurde.
[3] Qt ist eine C++-Klassenbibliothek für die plattformübergreifende Programmierung grafischer Benutzeroberflächen. Außerdem bietet Qt umfangreiche Funktionen für verschiedene Betriebssysteme bzw. Grafikplattformen an. [10]

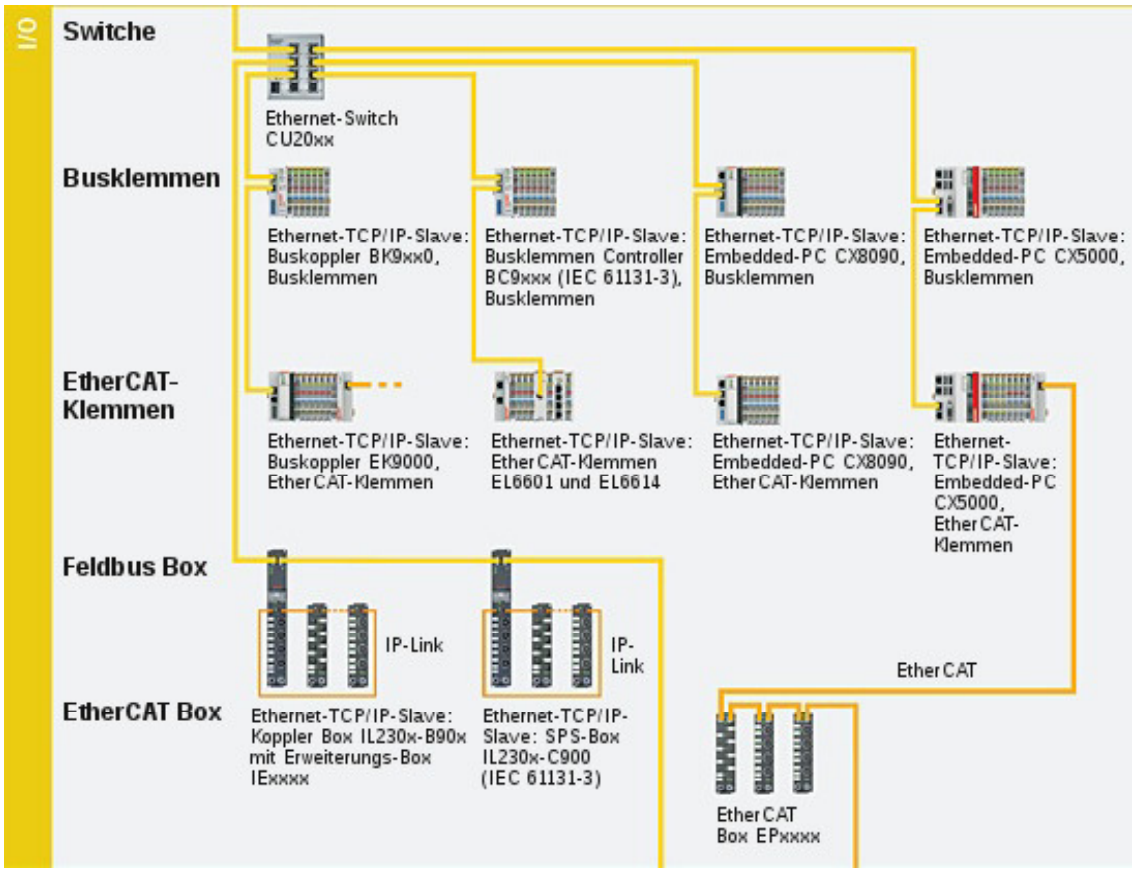

Abbildung 1.2: Übersicht Beckhoff Felbussystem [4]

den diese über das standardisierte Modbus-TCP Protokoll an das Leitsystem geschickt und dort gespeichert und angezeigt. Jede Steuerung hat ein eigenes Programm, welches speziell auf die individuellen Aufgaben zugeschnitten ist. Es sind zwei unterschiedliche Typen von Beckhoff-Steuerungen im Einsatz. Diese bieten unterschiedliche Möglichkeiten des Zugriffs auf Konfigurationseinstellungen und Prozessabbilder.

- PC basierte Steuerungen: Festlegung der Peripherie-Adressen und Umfangreiche Einstellmöglichkeiten über den Beckhoff TwinCAT-System-Manager.
- BC basierte Steuerungen: Eingeschränkte Einstellmöglichkeiten über Beckhoff KS-2000.

Mit der erfolgreichen Einführung der Feldebene für das Plansee-Meldesystem wurden die Grundsteine für ein langfristig erfolgreiches Prozessvisualisierungs- und Steuerungssystem gelegt. Nun ist es dem Instandhaltungspersonal einfach möglich das System selbständig zu pflegen, Diagnosen zu stellen und neue Signale oder neue Hardware zu integrieren. Verschiedene Funktionalitäten werden in Form von Modulen bereitgestellt. Durch die geschickte Kombination der zur Verfügung stehenden Module können viele Problemstellungen einfach und effizient gelöst werden. Zudem wird durch die zunehmende Standardisierung und die immer bessere Code-Qualität der Module die Gesamtverfügbarkeit des Systems gesteigert. Um die hohe Zuverlässigkeit und vor allem

die Wartbarkeit zu gewährleisten, sind sinnvolle Entwicklungskonzepte und -Vorgaben unumgänglich.

1.3.2 Konzept der Systementwicklung

Ein sehr wichtiger Teil des Plansee-Meldesystems ist das Entwicklungskonzept. Hierbei wurde, im Zuge der Einführung der verteilten Feldebene für das Plansee-Meldesystem, ein umfassendes Regelwerk zur Code-Erstellung und eine große Standard-Bibliothek mit vielen Funktionen und Panels geschaffen. Dieses Regelwerk umfasst folgende Hauptpunkte:

- Verpflichtender Coding-Style für alle Entwickler
- Code Dokumentation mit Hilfe eines eigens entwickelten Parsers [4] für das Dokumentationswerkzeug Doxygen
- Moduldokumentation im Wiki des Projektverwaltungssystemes
- Vorgeschriebener Testablauf mit Unit- und Integrationtests
- Automatisierte Tests über eigenen Testserver
- Einführung der Versionsverwaltung Git
- Fixe Release-Zeitpunkte mit entwicklungs- und stabilen Zweigen in der Versionsverwaltung

Das strukturierte Vorgehen bei der Systementwicklung hat eine merkbare Verbesserung der Codequalität zur Folge. Es treten viel weniger Fehler im Code auf. Kritische Bugs, welche die Stabilität des Gesamtsystems gefährden, hatten wir seit der Einführung des Entwicklungssystems gar nicht mehr. Durch diese Maßnahmen steigt auch die Akzeptanz bei den Anwendern und auch bei den Führungskräften. Da die Zuverlässigkeit des Systems steigt, steigt auch das Vertrauen in das System. Durch festgelegte Rahmenbedingungen bei der Entwicklung wird außerdem die Einarbeitungszeit neuer Entwickler verkürzt. Es wird versucht über Codereviews den Überblick aller Entwickler über Neuerungen und Änderungen zu verbessern. Die durchgängige Einführung und weitere Verbesserung der Entwicklungsmethoden wird auch einen wesentlichen Teil der vorliegenden Arbeit darstellen.

Der Aufbau und das Konzept zur Entwicklung des Plansee-Meldesystems wurde dabei weitgehend vom JCOP-Framework des Kernforschungszentrums Cern übernommen (siehe auch Abschnitt 2.3.2). Zum Teil wurden auch Funktionalitäten aus JCOP übernommen, indem Teile von JCOP in das Plansee-Meldesystem integriert wurden.

[4] „Ein Parser ist ein Computerprogramm, das in der Informatik für die Zerlegung und Umwandlung einer beliebigen Eingabe in ein für die Weiterverarbeitung brauchbares Format zuständig ist. Häufig werden Parser eingesetzt, um im Anschluss an den Analysevorgang die Semantik der Eingabe zu erschließen und daraufhin Aktionen durchzuführen.“ [20]

1.4 Ziel des Projektes

Ziel des Projektes ist die Konzeption und Erstellung einer Objektlaufzeit für flexible und dynamische Softwarekomponenten. Die Laufzeit soll mit den Möglichkeiten der PLC [5] in der jeweiligen Entwicklungsumgebung des Herstellers erstellt werden. Somit soll, zusätzlich zur direkten Abfrage der Prozessabbilder, eine Laufzeitumgebung auf Basis IEC-1131 Softwareobjekte aus dem Leitsystem aufnehmen und ablaufen lassen. Diese Objekte können alle denkbaren Funktionalitäten beinhalten (z.B. Zähler, Regelfunktionen, usw.) und Daten an das Leitsystem zurückliefern. Die Objekte können je nach Anforderung direkt auf der SPS oder auch im Leitsystem ablaufen. Die Laufzeit soll ausschließlich IEC-1131 Funktionen verwenden und ist damit auf allen gängigen Steuerungen lauffähig.

Die Objektlaufzeit führt Code direkt auf der dezentralen Peripherie aus. Damit soll es möglich sein, Software-Objekte unabhängig vom Leitsystem direkt auf der Steuerung ablaufen zu lassen. Das Spektrum der Aufgaben geht von einfachen Impulszählern bis hin zu komplexen Kommunikationsprozessen, beispielsweise mit Brandmeldeanlagen. Folgende Kriterien sollen dabei beachtet werden:

- Ablauf von vordefinierten Code-Objekte auf einer SPS oder im Leitsystem
- komplette Konfiguration der Objekte im Leitsystem
- das Laufzeitsystem wird vollständig im Beckhoff TwinCAT Programmiersystem erstellt
- das Laufzeitsystem stellt das Betriebssystem für die Objekte dar

Durch die Verwendung der Programmierumgebung des jeweiligen SPS-Herstellers, wird das System einfach auf andere Steuerungssysteme übertragbar. Das Laufzeitsystem stellt dabei eine Art Betriebssystem auf der SPS dar und ist eine Abstraktionsebene für den Ablauf der hardwareunabhängigen Software-Objekte. So soll eine sehr einfache Portierung auf andere Plattformen möglich sein. Objekte, zur Ausführung spezifischer Aufgaben, können mithilfe der Entwicklungsumgebung des jeweiligen Zielsystems sehr einfach bearbeitet oder ergänzt werden. Das Laufzeitsystem kümmert sich dabei um:

- die Speicherverwaltung aller Objekte
- die Bereitstellung persistenter Daten
- den Zugriffsschutz auf die Speicherbereiche
- den Datenaustausch mit dem Leitsystem
- die Konfigurationsschnittstelle aller Objekte
- die Diagnose und Alarmierung

[5] Eine speicherprogrammierbare Steuerung (SPS, englisch: Programmable Logic Controller, PLC) ist ein Gerät, das zur Steuerung oder Regelung einer Maschine oder Anlage eingesetzt und über ein Programm gesteuert wird. [12]

Ein derartiges Laufzeitsystem auf Basis Beckhoff BC9000 und CX9000 soll im Zuge der vorliegenden Diplomarbeit realisiert werden. Die Vorarbeiten wurden in der Arbeit zum Praxisprojekt II mit dem Thema *„Verteilte Feldebene für das Plansee Meldesystem“* beschrieben. [23]

Durch diese Erweiterung des Plansee-Meldesystems wird es möglich, benutzerdefinierten Code direkt in den lokalen CPUs der Feldsysteme auszuführen. Dies ermöglicht kürzere Reaktionszeiten und verteile Datenverarbeitung unabhängig vom Leitsystem. Da sowohl Laufzeitsystem und Softwareobjekte auf der SPS mit den Entwicklungswerkzeugen der jeweiligen Hersteller erstellt werden sollen, ist die Erweiterung der Funktionalitäten jederzeit möglich.

Es sollen folgende Ziele und Randbedingungen umgesetzt werden:

- Laufzeitsystem auf Basis von IEC 61131, erstellt mit der jeweiligen Entwicklungsumgebung des Hardware-Herstellers
- hardwareunabhängige Softwareobjekte für autarke Programmabarbeitung durch die jeweilige PLC
- Bibliothek mit Standard-Objekten und Beispielen für Siemens S7 und Beckhoff TwinCAT
- einfache Parametrierung der Softwareobjekte aus dem Leitsystem
- Programmerstellung, durch Zusammenschaltung von Softwareobjekten, direkt aus dem Leitsystem

2 Stand der Technik

Am Markt gibt es Systeme, die Leitsystem- und Feldebene miteinander verknüpfen. Allerdings sind diese Systeme herstellerspezifisch aufgebaut. Andere Systeme haben eine proprietäre Laufzeit, welche auf einem Industrie-Computer läuft. Diese Laufzeit kommuniziert dann über Feldbusprotokolle mit der Peripherie.

2.1 Normierung Speicherprogrammierbarer Steuerungen

„Die Europäische Norm EN 61131, die auf der internationalen Norm IEC 61131 basiert, befasst sich mit den Grundlagen Speicherprogrammierbarer-Steuerungen. Eine objektorientierte Weiterentwicklung für verteilte Steuerungen ist die EN 61499.“ [17]

Die Norm ist in sieben Teile gegliedert, wobei Teil 3 (EN 61131-3:2003) die Programmiersprachen behandelt. Hier werden fünf Programmiersprachen für speicherprogrammierbare Steuerungen festgelegt. Es sind textbasierte und grafische Sprachen definiert, wobei Funktionen und Funktionsblöcke aus jeweils anderen Sprachen verwendet werden können. Je nach Programmiersystem müssen nicht alle Sprachen zur Verfügung stehen. Einige Hersteller folgen bei den Bezeichnungen der Sprachen nicht der Norm, halten sich aber weitgehend an den festgelegten Funktionsumfang. Ziel der Normierung ist es, dem Anwender die Programmierung von Steuerungen unterschiedlicher Hersteller zu erleichtern. Trotzdem unterscheiden sich die Werkzeuge zur Programmerstellung von Hersteller zu Hersteller oft sehr stark. Für die Hardware-Konfiguration ist dabei immer herstellerspezifisches Spezialwissen erforderlich. [17]

2.2 Herstellerspezifische Systeme

2.2.1 Visualisierungssysteme

Nahezu alle Hersteller bieten für ihre Hardware passende Entwicklungssysteme an. Oft gibt es zusätzliche Visualisierungen, welche entweder direkt am Zielsystem oder auf speziellen Komponenten ausgeführt werden können. Dazu werden meist einfache Visualisierungen mit begrenzten Leitsystem-Funktionen angeboten. Weitere Software-Module können zugekauft werden um die Standardfunktionalitäten zu erweitern. Alle diese Systeme haben jedoch eine Einschränkung gemeinsam, sie laufen nur auf der Hardware der jeweiligen Hersteller. Somit müssen Programme und Bibliotheken bei einer Portierung mit großem Aufwand überarbeitet werden. [3] [36]

2.2.2 CODESYS

CODESYS ist eine Software-Plattform mit Programmierwerkzeugen auf Basis IEC 61131-3. Das System umfasst die Themengebiete Programmierung, Feldbus- und Hardware-Konfiguration sowie Visualisierung und Motion-Control. Spezifische Erweiterungen können über Bibliotheken und Plug-Ins eingefügt werden. Außerdem werden weitere Schnittstellen, wie z.B. OPC, angeboten. Die CODESYS-Software-Suite wird von der 3S-Smart Software Solutions GmbH entwickelt und vertrieben. Hardware-Hersteller können CODESYS durch den Erwerb von Soft- oder Runtime-Lizenzen auf ihrer Plattform einsetzen. Einige Hersteller von programmierbaren Automatisierungskomponenten haben CODESYS implementiert und nutzen das Softwaresystem für die Programmierung und Parametrierung der Komponenten. Somit kann eine Software für die Programmierung und Konfiguration von Automatisierungssystemen verschiedenen Hersteller verwendet werden. Für Anwender ist die Nutzung des CODESYS Development Systems kostenlos. Obwohl das System für diverse Plattformen unterschiedlicher Hersteller verwendet werden kann, ist die Portierbarkeit der erstellten Software nicht in allen Fällen sichergestellt. Vor allem Konfigurationseinstellungen unterscheiden sich zwischen den Herstellern sehr stark, wodurch wieder spezielles Know How der Anwender notwendig wird. [1]

2.2.3 Integrierte Automatisierungssysteme

Integrierte Automatisierungssysteme bieten ein durchgängiges Engineering vom Leitsystem bis zur Steuerung. Dabei werden die Steuerungskomponenten mit einer eigenen Laufzeitumgebung ausgerüstet. Somit lassen sich die Steuerungskomponenten im Leitsystem parametrieren und programmieren. Die Steuerungssoftware und die Visualisierung können durchgängig erstellt werden. Zusätzliche SCADA[6]-Funktionalitäten machen das System zu einem kompletten Leitsystem inklusive Datenaufzeichnung.

Eine Vielzahl integrierter Treiber und Protokolle ermöglichen die Anbindung unterschiedlichster Automatisierungshardware. Zum Teil kann die Laufzeitumgebung direkt darauf installiert und ausgeführt werden. Durch offene Protokolle können diese Produkte auch an übergeordnete Systeme gekoppelt werden und somit als alternatives Entwicklungssystem für Steuerungen verwendet werden. Es lassen sich eigene Bausteine in Hochsprachen entwickeln. Diese können in Bibliotheken hinterlegt und grafisch verknüpft werden. Instandhaltungsebenen bieten Diagnose- und Handeingriffsfunktionen für Techniker. Die verwendeten Software-Bausteine haben bereits Visualisierungsdaten hinterlegt, wodurch einfach Bedienoberflächen erstellt werden können. Die Grafikfunktionen bieten Scripting-Möglichkeiten und weitreichenden Funktionalitäten für komplexe Anlagenbilder. Diese können auch direkt in den angebotenen SCADA-Erweiterungen

[6] Unter Supervisory Control and Data Acquisition (SCADA) versteht man das Überwachen und Steuern technischer Prozesse mittels eines Computer-Systems. [13]

der Systeme verwendet werden. Die Systeme verstehen sich als Plattformen und sind für Erweiterungen offen.

Die PLC wird bei diesen Automatisierungssystemen durch eine Laufzeitumgebung ersetzt, welche auf einem Industrie-PC (IPC) oder einer entsprechenden Steuerung mit Betriebssystem läuft. Dadurch ist das vom Hardware-Hersteller zertifizierte System nicht mehr vorhanden. Dies kann zu Verlust der Echtzeiteigenschaften oder reduzierter Sicherheitsfunktionen führen. Zur Ausführung der Laufzeit ist eine entsprechende Hardware Plattform (IPC oä.) notwendig. Daher kann die Ausführung auf einer Steuerung ohne entsprechendes Betriebssystem nicht erfolgen.

Der Großteil der bei Plansee eingesetzten Steuerungen sind aber Typen ohne Betriebssystem. Abhilfe könnten nur zentrale IPCs bieten, die aber wiederum zusätzlichen Aufwand und Fehlerquellen bedeuten. Daher soll eine plattformunabhängige SPS-Laufzeit umgesetzt werden. Diese würde den Einsatz auf allen Steuerungstypen ermöglichen und ebenfalls einen hohen Komfort bei der Parametrierung der Funktionalitäten bieten. [25] [33]

2.3 Offene Frameworks

2.3.1 UNICOS

UNICOS (UNified Industrial Control System) ist ein am CERN[7] entwickeltes industrielles Steuerungs- und Kontrollsystem. Dabei werden Vorgehensweisen und Methoden für das Design und die Entwicklung von SCADA- und PLC-Anwendungen standardisiert. UNICOS verwendet, wie auch das Plansee-Meldesystem, WinCC OA als SCADA-Plattform. Derzeit sind PLCs der Firmen Siemens und Schneider über UNICOS angebunden. Außerdem wird UNICOS für Systeme mit Codesys, Industrie-PC-Plattformen und spezielle Anwendungen des Teilchenbeschleunigers verwendet. Das UNICOS-Framework stellt außerdem eine große Zahl an Standard-Gerätetypen, als sogenannte Objekte, bereit. Diese Objekte werden sowohl für die Kontrolle und Steuerung der Systeme, als auch für die Modellierung verwendet. Dabei reichen die Modelle von der Programmierung logischer Einheiten bis hin zur Steuerung von Prozessabläufen. Hierbei geht es nicht nur um die einzelnen Geräte sondern um den gesamten Entwick-

[7] Am CERN, der Europäischen Organisation für Kernforschung, sind Physiker und Ingenieure mit der Erforschung der grundlegenden Struktur des Universums beschäftigt. Sie nutzen die weltweit größten und komplexesten wissenschaftlichen Instrumente, um die grundlegenden Bestandteile der Materie zu studieren - die fundamentalen Teilchen. Bei der Kollision der Teilchen, mit beinahe Lichtgeschwindigkeit, erhalten die Physiker Hinweise, wie die Teilchen in Wechselwirkung treten und bekommen so Einblicke in die grundlegenden Gesetze der Natur. Die am CERN verwendeten Instrumente sind spezielle Teilchenbeschleuniger und Detektoren. Diese beschleunigen die Teilchenstrahlen auf hohe Energien, um miteinander oder mit stationären Zielen kollidieren. Dabei beobachten die Detektoren dieser Kollisionen und zeichnen die Ergebnisse auf. [7]

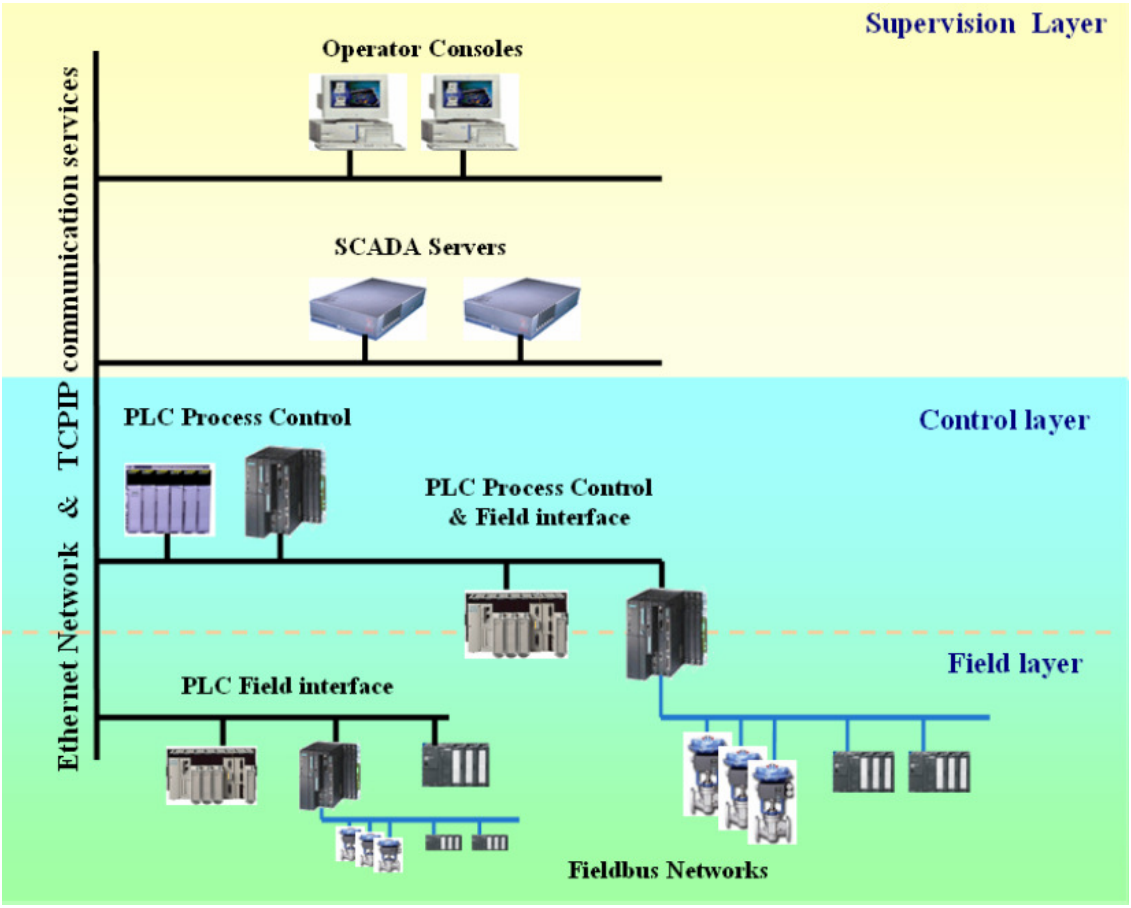

Abbildung 2.1: industrielles Steuerungssystem in drei Schichten [28]

lungsprozess von Überwachungs- und Steuerungssystemen. Dieser Entwicklungsprozess beinhaltet folgende Schritte:

- Bereitstellung von Werkzeugen für die schnelle Entwicklung von Steuerungsanwendungen, mit teilweise automatischer Codegenerierung
- Werkzeuge für Inbetriebnahme und Tests: automatische Testroutinen, Konfigurations-Oberflächen für Geräte- und Kommunikations-Tests
- Module für den Betrieb: standardisierte Benutzeroberflächen, Navigationskonzepte, Alarmhandling und Berechtigungsverwaltung für Benutzer
- Systemdiagnose: Übersichtspanels, Alarme und Ereignisse, automatische Diagnose-Routinen
- Wartung und Instandhaltung: standardisierte Fehlerbehandlung und Wartungsunterstützung für Instandhaltungspersonal

UNICOS wurde für die Erstellung von dreischichtigen industriellen Steuerungssystemen entwickelt (siehe Abbildung 2.1). Es soll die volle Funktionalität der Prozesse im überlagerten Steuerungssystem abgebildet werden. Die Modellierung erfolgt hierarchisch und folgt der Struktur der Objekte. Das sind I/Os, tatsächliche Geräten und abstrakteren Steuer-Objekte. Diese Objekte werden als gemeinsame Sprache zwischen Prozess-

ingenieure und Programmierer verwendet und dienen der funktionellen Analyse von Prozessen. Zusätzlich zu diesem Verfahren wurden Werkzeuge entwickelt um die Instanziierung der Objekte und die automatische Generierung von Programmgerüsten für die Steuerungen zu unterstützen. Die Systemgenerierung folgt dabei einem standardisierten Ablauf, welcher in vier Schritte unterteilt ist.

1. Erstellung von Spezifikationen: Logik und Funktionalitäten werden als Textbausteine in speziellen Templates definiert. Daten und Instanzen werden in strukturierten Listen gepflegt.
2. Automatische Generierung (UAB - UNICOS Application Builder): Erzeugung von Instanzen und Standardlogiken aus den Spezifikationen.
3. Manuelle Anpassungen: Kontrolle und Anpassung der automatisch erzeugen Ressourcen und Abläufe. Nachdem alle Funktionen getestet wurden, können die Änderungen auf die Zielsysteme geladen werden.
4. Dokumentation: Dokumentationen der Änderungen werden teilweise automatisch erstellt und müssen noch korrigiert oder angepasst werden.

Das UNICOS-Framework besteht aus unterschiedlichen Paketen welche teilweise auf dem JCOP-Framework basieren (siehe Abschnitt 2.3.2). [8] [28]

2.3.2 JCOP

Das Joint Controls Project (JCOP) des CERN stellt eine Sammlung von Richtlinien und Werkzeugen für die Entwicklung eines industriellen Überwachungs- und Steuerungssystems bereit. Auch Projektabläufe und Vorgehensweisen zu Erweiterungen und Fehlerbehebungen sind in JCOP definiert. JCOP soll die Entwickler des Control-Layers, eine dreistufigen SCADA-Systems (Abbildung 2.1), bei ihrer Arbeit unterstützen. Zusätzlich sind Trainingsmaterialien für die Ausbildung von Entwicklern und Anwendern enthalten. Folgende Richtlinien für die Entwicklung von Software und Panels [8] sind in JCOP festgelegt:

- Anwendungshinweise zum Framework
- Allgemeine Richtlinien und Konventionen
- Entwicklungsrichtlinien (Coding Style)
- Namenskonventionen
- Richtlinien zur Grafikerstellung
- Richtlinien zur Softwareentwicklung
- Modellierung von Geräten

Das JCOP-Framework enthält eine Vielzahl unterschiedlicher Standard-Komponenten welche fertige Funktionalitäten oder Teilfunktionen bereitstellen. Diese Komponenten

[8] grafische Anzeigefenster des Leitsystems

können mittels eines Installationswerkzeugs zu den Projekten hinzugefügt, entfernt oder aktualisiert werden. [30] [29]

2.4 Forth

„Forth ist eine imperative, Stack basierte Programmiersprache. Ein Forth-System beinhaltet ein Betriebssystem zum Ablauf und eine Entwicklungsumgebung zur Erstellung von Forth-Programmen.“ [18] Forth wird oft für einfache Steuerungsanwendungen verwendet, kommt aber auch in komplexen Applikationen zum Einsatz. Forth wurde 1969 von Charles Moore als Echtzeitprogrammiersprache für die Steuerung von Radioteleskopen entworfen.

Forth basiert auf dem Konzept einer Zwei-Stackmaschine. Jegliche Datenverarbeitung wird auf dem sogenannten Datenstack ausgeführt. Der Datenstack dient zur Parameterübergabe zwischen Operatoren. Der zweite Stack, Returnstack genannt, speichert die Rückkehrinformationen bei verschachteltem Aufruf von Unterprogrammen. Der Quellcode besteht aus einzelnen Sequenzen, die durch einen Compiler zu Unterprogrammen, sogenannten Worten, übersetzt werden. Neu erstellte Worte erweitern den verfügbaren elementaren Wortschatz. Intern werden Forth-Worte in einem Wörterbuch (Dictionary) verwaltet. Für die Reihenfolge von Parametern und Operatoren bei der Programmerstellung verwendet Forth eine Postfix-Notation [9] und bietet eine interaktive Arbeitsweise bei der Programmentwicklung und bei Tests. Alle Worte lassen sich über den Stack mit Parametern versehen und über einen Interpreter aufrufen. Die Ergebnisse werden dann interaktiv vom Interpreter zurückgegeben. [26] [6]

Forth hat viele einzigartige Eigenschaften und Anwendungen [27]:

- Erweiterbar, durch neue Wörter
- Äußerst effiziente Thread-basierte Sprache
- Interaktive Entwicklung und Test
- Einfach auf andere Systeme portierbar
- Ressourcen schonend, sehr kleine Codebasis

2.5 BACnet

„BACnet (Building Automation and Control Networks) ist ein Netzwerkprotokoll für die Gebäudeautomation.“ [14] BACnet soll die Kommunikation und den Datenaustausch

[9] „Die umgekehrte polnische Notation (UPN) oder reverse polnische Notation (englisch reverse Polish notation, kurz RPN), auch Postfixnotation genannt, ist eine von der polnischen Notation abgeleitete Schreibweise bzw. Eingabelogik für die Anwendung von Operationen. Bei der umgekehrten polnischen Notation werden zunächst die Operanden niedergeschrieben bzw. eingegeben und danach der darauf anzuwendende Operator.“ [22]

zwischen Geräten unterschiedlicher Hersteller gewährleisten. Dazu werden in der Norm Dienste und Objekttypen definiert. Als Objekttypen sind beispielsweise Geräte, Analog-Ein und Analog-Ausgänge, Zähler, Trend-Objekte usw. definiert. BACnet wird weltweit für die standardisierte Datenkommunikation in der Gebäudeautomation verwendet. Es können definierte BACnet Objekte zwischen den Geräten ausgetauscht werden. Die Objekte und Funktionalitäten werden dabei in einem Programm definiert und in die Geräte geladen. Mittels BACnet können dann die Parameter der einzelnen BACnet Objekte manipuliert und eingestellt werden. Dabei ist es nicht möglich die BACnet Objekte selbst zu manipulieren, Objekte zu löschen oder zu erzeugen.

2.6 Betriebliche Situation

Durch die Positionierung des Plansee-Meldesystem als zentrales Prozessvisualisierungs- und Steuerungssystem am Standort Reutte werden laufend weitere Subsysteme integriert. Auch die Zahl der Benutzer erhöht sich ständig. Um auch weiterhin mit den neuen Herausforderungen und den wachsenden Anforderungen Schritt halten zu können, sind weitere Maßnahmen notwendig.

2.6.1 Bestehende Systemlandschaft

Die Vorstellung des derzeitigen Systems bei den Entscheidungsträgern hat gezeigt, dass eine Homogenisierung der Systemlandschaft gewünscht ist. Derzeit sind fünf verschiedene Leitsysteme am Standort Reutte im Einsatz.

- Plansee-Meldesystem: Basis ist das SCADA-System Siemens WinCC OA. Wird für die Steuerung und Überwachung aller Systeme und Anlagen im Bereich der Infrastruktur und Medienversorgung eingesetzt.
- Wonderware-Intouch: Wurde in den Prozessanlagen der alten Generation eingesetzt. Neue Anlagen werden bereits mit WinCC OA ausgestattet.
- Siemens-Desigo Insight: Dieses System dient zur Steuerung und Überwachung der Heizungs-, Lüftungs- und Klimaanlagen (HLK) am Standort.
- Siemens-VISONIK: Wird für die HLK-Anlagen der älteren Generation verwendet und ist Teil des Desigo Insight Systems
- Citect-Exomatic: Steuert und überwacht ebenfalls einen Teil der HLK-Anlagen.

Diese Vielfalt an unterschiedlichen Systemen bringt einige Probleme bei der Betreuung mit sich. Techniker und Instandhaltungspersonal müssen auf allen Systemen ausgebildet sein um Wartungen und Reparaturen durchführen zu können. Gerade für die Mitarbeiter im Bereitschaftsdienst stellt dies eine enorme Herausforderung dar. Einige der Systeme werden von den Herstellern nicht mehr weiterentwickelt. Dies hat zur Folge, dass Änderungen und Anpassungen oft nicht mehr möglich sind. Um in Zukunft

Synergien bei der Erstellung, Anpassung und Betreuung der oben genannten Systeme nutzen zu können sollen diese mittelfristig auf eine gemeinsame Basis gestellt werden. Diese Basis wird WinCC OA sein, da mittels der verteilten Systeme Daten und Ressourcen gemeinsam genutzt werden können. Somit funktionieren die Systeme der Medienversorgung und der HLK weiterhin eigenständig, teilen sich aber die gemeinsamen Kommunikationsnetzwerke.

Im Bereich der Heizungs-, Lüftungs- und Klimatechnik müssen Teile der Hardware ohnehin getauscht werden, da keine Ersatzteile mehr erhältlich sind. Hier soll in Zukunft auf den BACnet Standard (Abschnitt 2.5) gesetzt werden. Die Anbindung an WinCC OA erfolgt dann über den BACnet-Treiber welcher als lizenzpflichtiges Modul erhältlich ist. Um den Aufwand der Systemumstellung und die zu erwartenden Kosten besser einschätzen zu können, soll ein Prototyp mit einer BACnet fähigen HLK-Anlage erstellt werden. Da die Kommunikationsschnittstelle zugekauft wird, sind Anpassungen lediglich im Bereich der Visualisierung zu erwarten. Auch die Kombination von BACnet Objekten zu komplexeren Bausteinen stellt eine Herausforderung dar. Hier steht allerdings ein erheblicher finanzieller Aufwand dahinter, da auch die Systeme und Anlagen entsprechen angepasst werden müssen. Zusätzlich sind personelle Ressourcen und Mittel für Schulungen bereitzustellen.

Um das mittelfristige Ziel zur Vereinheitlichung der Systeme zu fixieren, wurden Entwicklungsmeilensteinen vereinbart. Dabei wurde, unter Anderem, die Positionierung des Plansee-Meldesystems als zentrales Prozessvisualisierungs- und Steuerungssystem fixiert.

2.6.2 Erweiterung des Plansee-Meldesystems

Bei der Erweiterung des Plansee-Meldesystems soll das Augenmerk auf die weitere Vereinfachung des Betriebsablaufes gelegt werden. Primäres Ziel ist allerdings die Systemsicherheit und die Absicherung der Verfügbarkeit aller Teilsysteme. Dies soll durch neue Entwicklungsmethoden und standardisierte Module erreicht werden. Einige der bereitgestellten Module sind vom Design des User Interface nicht optimal. Benutzer finden sich oft nur schwer zurecht, da Aufbau und Menüführung uneinheitlich sind. Um diese Mängel zu beheben wird derzeit das Design-Konzept überarbeitet und standardisiert. In Zukunft soll es somit einen einheitlichen Aufbau aller Anzeige- und Eingabemasken im System geben. Auch der Aufbau der Software-Module soll in einem Framework weiter standardisiert werden.

Außerdem steht auch die Integration von weiteren Gewerken an. Kürzlich wurden Module zur Ansteuerung von seriellen Störungsanzeigen, für die Kopplung mit dem Zutrittssystem und für die Anbindung der Einbruchmeldeanlage erstellt und implementiert. Zur Absicherung der hohen Verfügbarkeit des Systems wurden alle kritischen System-

komponenten in den redundanten Betrieb eingegliedert. Damit ist nun sichergestellt, dass beim Ausfall eines Servers eine stoßfreie automatische Umschaltung auf den redundanten Partner erfolgt. Alle Systemprozesse sind im Normalbetrieb auf beide Server verbunden, sodass deren Funktion durch einen Ausfall eines Partners nicht beeinflusst wird.

Um die Wiederverwendung von Softwaremodulen zu vereinfachen wurde das Planseе-Meldesystem-Projekt in zwei Subprojekte aufgespalten. Es wurden alle übergreifenden Funktionalitäten in das Core-Subprojekt ausgelagert. Im Core sind nun alle Module des Frameworks enthalten, wobei projektspezifische Implementierungen weiterhin möglich sind. Solche Anpassungen dabei direkt im Hauptprojekt durchgeführt. Durch die Subprojekt-Funktionalität von WinCC OA können beliebig viele Subprojekte in ein Hauptprojekt eingehängt werden. Dabei werden die Dateien aller Subprojekt beim Laden des Hauptprojektes eingebunden. Somit können alle Funktionen und Panels der Subprojekte im Hauptprojekt direkt verwendet werden. Außerdem gibt es die Möglichkeit Dateien aus den Subprojekten im Hauptprojekt zu überschreiben, sodass die jeweiligen Funktionalitäten projektspezifisch angepasst werden können. Durch die Aufteilung in Hauptprojekt und Subprojekte ist es nun möglich die Kernfunktionalitäten auszutauschen oder einfach mit anderen Projekten zu teilen. Am Standort Reutte kann die Kernbibliothek über Abteilungsgrenzen hinweg von anderen WinCC OA Entwicklern verwendet werden. Außerdem können diese Kernbibliotheken von externen Entwicklern verwendet und erweitert werden. Diese Vorgehensweise fördert die Wiederverwendung und vereinfacht die Arbeit mit ständig neuen Modulen.

2.6.3 Einbinden der Systembenutzer

Um die Einarbeitung von Benutzern aber auch von Entwicklern zu erleichtern und zu beschleunigen wird eine Vereinfachung der Benutzeroberfläche angestrebt. Dazu wird das Design und Bedienkonzept standardisiert und dokumentiert und muss in Zukunft für alle Module einheitlich angewendet werden. Außerdem werden die Dokumentationen für die bestehenden Module angepasst und überarbeitet. Über das zentrale Wiki des Projektmanagementsystems werden diese Dokumentationen allen Benutzern zur Verfügung gestellt. Außerdem können allen Benutzer über das Ticketsystem Fehler melden oder sinnvolle Erweiterungen und Verbesserungen anregen.

Zur Absicherung des Systems und für die einfache Freigabe von Modulen wurde ein Berechtigungskonzept ausgearbeitet und in Form eines Software-Moduls umgesetzt. Nun ist es möglich Berechtigungen für jedes einzelne Modul zu vergeben. Zusätzlich gibt es für alle Module einheitliche Berechtigungsstufen welche den Benutzergruppen individuell zugewiesen werden können. Durch diese feingranulare Steuerung der Berechtigungen sollen Fehlbedienungen und ungewollte Änderungen weitestgehend vermieden werden. Außerdem bekommen die Mitarbeiter nur jene Module freigeschalten für die sie

berechtigt und geschult sind.

Um ein System zu etablieren und die sichere Verwendung zu gewährleisten, müssen die Benutzer geschult werden. Dabei ist auch die Erstellung von Schulungspaketen wichtig. Die Mitarbeiter sollen in Zukunft fachspezifisch auf die einzelnen Module ausgebildet werden. Dazu gehören sowohl ordentliche Dokumentationen und Schulungsunterlagen, aber auch Anleitungen und Tutorien. Die Schulungspakete erleichtern die regelmäßigen praktischen Übungen und Weiterbildungen und gewährleisten die Durchgängigkeit des Schulungsprogrammes. Durch die regelmäßigen Erweiterungen und die kurzen Release-Zyklen werden auch die Funktionen der Module und die Bedienoberflächen häufige geändert. Um die Anwender darüber auf dem Laufenden zu halten sind eben diese regelmäßigen Schulungen unumgänglich. Es wird daher versucht, wöchentliche Praxis-Schulungen von circa einer halben Stunde durchzuführen. Dabei sollen die Fachbereiche individuell auf den Funktionalitäten den Plansee-Meldesystems ausgebildet werden. Bei jeder Schulung ist es wichtig, dass die Inhalte auf die Teilnehmer abgestimmt sind. Ansonsten ist die Schulung nicht zielführend.

Weiters stellt das Thema personelle Ressourcen im Entwicklungsteam eine erhebliche Herausforderung dar. Da es nur einen internen Kernentwickler gibt, müssen viele Entwicklungsaufgaben an externe Dienstleister ausgelagert werden. Auch die Betreuung und der Kontakt zu den Benutzern kommen dadurch zu kurz. Zwar helfen die Vorgaben und Standardisierungen, sowie die Testprozeduren, die Qualität des Codes zu stabilisieren. Trotzdem ist ein erheblicher Aufwand für die Betreuung der externen Mitarbeiter notwendig. Durch gegenseitige Reviews wird versucht die Einhaltung der Vorgaben zu überwachen. Außerdem bringen diese Reviews auch eine Möglichkeit des Informationsaustausches zu den Änderungen mit sich. Allerdings kosten diese Koordinationstätigkeiten auch viel Zeit, welche dann wieder bei der Systementwicklung fehlt. In Zukunft soll deshalb eine stärkere Trennung zwischen Kern- und Projektentwicklung stattfinden. Durch die Auftrennung des Gesamtprojektes in Subprojekte können die Entwicklungsaufgaben leicht aufgeteilt werden. So sollen weitere Entwickler aus den Fachabteilungen die Möglichkeit bekommen sich an der Weiterentwicklung des Systems zu beteiligen.

3 Präzisierung der Aufgabenstellung

Als Erweiterung des Plansee-Meldesystems sollen logische Verknüpfungen aber auch komplexe Funktionen nicht nur in der Leitsystemebene sondern auch direkt in den dezentralen Speicherprogrammierbaren-Steuerungen ablaufen können. Dazu werden die Funktionalitäten im Leitsystem definiert und mit Ein- und Ausgängen oder Variablen verknüpft. Anschließend kann diese Logik wahlweise im Leitsystem oder direkt auf einer Steuerung im Feld ausgeführt werden.

Bisher könnten logische Verknüpfungen und Funktionalitäten nur auf der Leitsystemebene realisiert werden. In der vorliegenden Arbeit wird ein Laufzeitsystem entwickelt, welches die einfache Erstellung von Funktionalitäten auf den Steuerungen unterstützt. Diese Erweiterung bietet die Möglichkeit, Funktionen direkt in einer Steuerung abzuarbeiten. Dadurch werden diese unabhängig vom übergeordneten Kommunikationsnetzwerk und vom Leitsystem. Sicherheit und Bedienbarkeit werden erhöht und der Implementierungsaufwand für neue Funktionalitäten gesenkt.

3.1 Nutzer des Systems

Das hier beschriebene Laufzeitsystem für Software-Objekte betrifft nur Systemtechniker, Administratoren und Systementwickler. Die Systembenutzer sollten im Idealfall von den Neuerungen nichts mitbekommen. Es sind allerdings indirekte Auswirkungen zu erwarten, da durch das Laufzeitsystem Ausfälle einzelner Steuerungen bei Programmänderungen ausbleiben.

3.1.1 Bediener in den Leitwarten

Wie bisher, sollen die jeweilige Implementierungen von Funktionen, sowie die Datenherkunft für die Bediener des Systems transparent sein. Die Bediener in den Leitwarten sollen nur die wichtigen und notwendigen Informationen für die Beurteilung der jeweiligen Situation bekommen. Besonders wichtig ist dabei, dass die Bediener nicht mit verwirrenden und irreführenden Bezeichnungen konfrontiert werden. Vom hier beschriebenen Laufzeitsystem und von den Software Objekten sollen die Bediener nichts mitbekommen, da für Sie die Herkunft der Informationen vollkommen belanglos ist.

3.1.2 Systemtechniker

Die Systemtechniker sind die eigentlichen Bediener des Laufzeitsystems. Sie verknüpfen die Software-Objekte und Ein- und Ausgänge zu Funktionalitäten und verteilen die-

se auf die Ausführungseinheiten. Durch das durchgängige Engineering soll der Aufwand von Anpassungen und Erweiterungen verringert und die Bedienung der Werkzeuge erleichtert werden. Bisher mussten für die Konfigurationen im Leitsystem und an den Speicherprogrammierbaren-Steuerungen unterschiedliche Software-Tools verwendet werden, was die Techniker teilweise vor große Herausforderungen stellte. Bei der Entwicklung des Laufzeitsystems wird es eine enge Zusammenarbeit zwischen Entwicklern und Systemtechnikern geben. Vor allem für die Techniker soll das neue System die Arbeit erleichtern und ihnen die Möglichkeit bieten eigenständig Funktionalitäten auf den Steuerungen zu implementieren.

3.1.3 Datenpfleger

Im Leitsystem werden die Daten in sogenannten Datenpunkten abgelegt. Die Datenherkunft ist für die Systempfleger unerheblich und nicht sichtbar. Somit betreffen die Änderungen und Erweiterungen die Datenpfleger nicht.

3.1.4 Administratoren

Die Administratoren sind für den Betrieb des Gesamtsystems aber hauptsächlich für das Leitsystem zuständig. Ziel ist es, die Softwareänderungen durch die im Kapitel 1.3.2 beschriebenen Maßnahmen entsprechen abzusichern. Für die Administration, Diagnose und Fehlerbehebung der Leitsystem-Module sind außerdem Schulungsmaßnahmen für Administratoren geplant. Somit sind die Administratoren erst bei der Implementierung der Änderungen betroffen und werden bei der Entwicklung nur am Rande eingebunden.

3.1.5 Systementwickler

Auf die Systementwickler kommt der Hauptteil der Aufgaben zu. In der vorliegenden Arbeit sollen das Laufzeitsystem und eine Standard-Bibliothek einiger Software-Objekte erstellt werden. Außerdem wird ein rudimentäres Interface für das Handling der Objekte im Leitsystem geschaffen. Die weiter Verfeinerung des User-Interface und zusätzliche Objekte sollen durch das Entwicklungsteam, bestehend aus einem firmeninternen und zwei externen Entwicklern, umgesetzt werden.

3.2 Nichtfunktionale und Funktionale Eigenschaften

Die Techniker müssen die Möglichkeit haben, im übergeordneten Leitsystem Funktionen zu definieren welche dezentral in den Speicherprogrammierbaren-Steuerungen abgearbeitet werden. Die Definition, Parametrierung und das Laden der Funktionen in

die Steuerungen sollen direkt aus dem Leitsystem erfolgen. Es soll ein durchgängiges Engineering von Leitsystem und Steuerung in einer einheitlichen Programmumgebung möglich sein. Um den Technikern einen möglichst einfachen Umgang mit dem Laufzeitsystem zu ermöglichen, soll besonders auf die Einfachheit und die intuitive Bedienung des Systems geachtet werden. Es soll einfach und ohne spezielle Kenntnisse möglich sein neue Objekte aus einem Objekt-Katalog auszuwählen und in das Laufzeitsystem zu integrieren.

Im ersten Schritt werden die Objekte in der Entwicklungsumgebung des Hardwareherstellers erstellt und auf die Steuerung geladen. Sind die Objekte einmal in Laufzeitsystem bekannt, können diese aus dem Leitsystem instanziiert und verknüpft werden. Aus dem vorangegangenen Projekt zur verteilten Feldebene wurden die Notwendigkeit zur Bereitstellung von Werkzeugen zur Diagnose und Fehlerbehebung deutlich. Deshalb muss die Laufzeitumgebung diese Diagnosemöglichkeiten bieten. Dabei sollen Informationen über die internen Zustände der Objekte und über die Verknüpfungen im Leitsystem angezeigt werden.

Folgende Anforderungen an das Laufzeitsystem für Software-Objekte sind besonders wichtig:

- durchgängige Parametrierung des Laufzeitsystems und der Objekte im Leitsystem
- Verknüpfung von Ein- und Ausgängen sowie Variablen mit Objekten
- einfache Handhabung und Erweiterbarkeit
- Objekte können sowohl im Leitsystem als auch auf den Steuerungen im Feld ablaufen
- Unterstützung bei Diagnose und Fehlerbehebung durch das Laufzeitsystem

3.3 Abgrenzung des Systems

Das Laufzeitsystem für Software-Objekte ist nur für kleinere Datenerfassungsaufgaben und Verknüpfungen mäßiger Komplexität konzipiert. Es ist nicht Ziel, die Automatisierung komplexer Abläufe oder ganzer Anlagen damit abzubilden. Weiters kann das System keine sicherheitskritischen Verknüpfungen handhaben und keine sicheren Abschaltungen durchführen. Da das Laufzeitsystem spezielle Anforderungen an die ausführbaren Objekte stellt, können Standardbausteine aus herkömmlichen Bibliotheken nicht verwendet werden. Diese müssen, wie bisher, direkt in der Entwicklungsumgebung des jeweiligen Steuerungsherstellers programmiert und auf die Steuerung geladen werden. Es ist möglich das Laufzeitsystem parallel mit herkömmlichen Programmteilen auf einer Steuerung zu betreiben. Hier sind allerdings die Einschränkungen der Hardware hinsichtlich Speicherplatz und Rechenleistung zu beachten.

In der geplanten Version des Systems müssen Objekte zum Ablauf im Laufzeitsystem des Leitsystems separat erstellt werden. Die Laufzeitsysteme auf den dezentralen Steuerungen im Feld und im Leitsystem sind nicht einheitlich. Für die weitere Entwicklung ist ein einheitliches Laufzeitsystem sowohl für das Leitsystem als auch für die Speicherprogrammierbaren-Steuerungen geplant. Dieses einheitliche Laufzeitsystem ist allerdings nicht Teil dieser Arbeit.

Das Laufzeitsystem wird für Zielsysteme der Firma Beckhoff mit der Entwicklungsumgebung TwinCAT 2 [10] erstellt. Ebenso werden die Objekte für die Beckhoff-Hardware in der genannten Umgebung entwickelt. Eine Erweiterung auf Hardware der Firma Siemens [11] ist geplant. Zielsysteme der Firmen Beckhoff und Siemens sollen so durch das einheitliche Laufzeitsystem für Leitsystem und Feldsteuerungen unterstützt werden. Grundsätzlich ist aber auch die Portierung auf weitere Steurungssysteme anderer Hersteller möglich.

Die vorliegende Arbeit umfasst nur einfache Panels zur Parametrierung und Diagnose des Laufzeitsystems. Es sind dialogbasierte Masken für die Parametrierung und textbasierte Logfiles für die Diagnose vorgesehen. Eine geplante Erweiterung des Systems beinhaltet allerdings eine erweiterte grafische Bedienoberfläche in der Verknüpfungen grafisch erstellt werden können. Der Diagnose-Modus soll dann die aktuellen Zustände der Objekte in der Grafik anzeigen.

3.4 Konkrete Implementierung

Um die Verwendung und die Funktionalität des Laufzeitsystems zu zeigen, wird im Zuge dieser Arbeit die Steuerung der Notstromaggregate am Standort Reutte im Planseer-Meldesystem implementiert. Dabei werden die Notstrom-Diesel für die Spitzenlastabdeckung genutzt, um ein möglichst konstantes Lastprofil zu erzeugen. Mehrere Aggregate werden im Netzbetrieb gestartet und speisen ihre elektrische Leistung in das Stromnetz ein. So wird der aktuelle Leistungsbezug vom Energieversorger gedämpft. Die Aggregate müssen dazu zum richtigen Zeitpunkt über eine Lastprognose gestartet werden. Ziel ist es, ein möglichst konstantes Band bei der Energieabnahme zu erzeugen. Es wird der Motor des Aggregates gestartet um eine Minute warm zu laufen. Nach der Warmlaufzeit wird ein Sollwert von achtzig Prozent der Gesamtleistung des Aggregates in einer Rampe angefahren. Wenn das Aggregat nicht mehr benötigt wird, wird der Sollwert in einer Rampe bis Null gefahren. Anschließend wird der Motor nach einer Abkühlzeit von zwei Minuten gestoppt. Außerdem werden diverse Störungen und die Betriebszustände überwacht.

Die Steuerung des Aggregates soll dabei autark in einer Speicherprogrammierbaren-

[10] http://www.beckhoff.at/default.asp?twincat/default.htm
[11] http://w3.siemens.com/mcms/simatic-controller-software/de/step7/seiten/default.aspx

Steuerung ablaufen. Aus dem Leitsystem werden lediglich die Befehle Start und Stopp an die Steuerung übermittelt. Zusätzlich soll auch eine Betriebsart vorgesehen werden, in der aus dem Leitsystem auch der Sollwert der Aggregate vorgegeben werden kann. Um diese Funktionalität auf den Steuerungen zu implementieren soll das neu entwickelte Laufzeitsystem verwendet werden. So können die Aggregate einfach in das Plansee-Meldesystem integriert und von den Systembenutzern aus dem System parametriert und diagnostiziert werden.

4 Systemkonzeption

Als bewährtes Leitsystemprodukt wird SIMATIC WinCC Open Architecture (WinCC OA) von ETM eingesetzt. In der Feldebene werden PC und BC basierte Steuerungen der Firma Beckhoff eingesetzt. Genauere Informationen zum bestehenden System finden sich im Kapitel 1.3.1.

4.1 Fixer Funktionsmix

Einige Hersteller von Speicherprogrammierbaren-Steuerungen bieten sogenannte Kompaktsteuerungen an. Diese sind meist mit einer definierten Peripherie ausgestattet, welche bereits in das Gerät integriert ist. Meist sind mehrere digitale und wenige analoge Ein- und Ausgänge verbaut. Bei manchen Systemen ist die Hardware-Konfiguration ab Werk vorgegeben. Bei Anderen kann sie vom Anwender angepasst oder erweitert werden.

In Anlehnung an dieses Prinzip besteht auch für das Laufzeitsystem die Möglichkeit vordefinierte Konfigurationen für die Steuerungen zur Verfügung zu stellen. Hierbei beinhaltet die Konfiguration bestimmte Funktionen welche fest mit der Peripherie verknüpft sind. Die festgelegten Funktionalitäten werden mit den Entwicklungswerkzeugen des jeweiligen Hardware-Herstellers entwickelt und auf das Zielsystem geladen. Dabei werden fixe Kommunikationskanäle für den Datenaustausch zwischen Steuerung und Leitsystem genutzt. Alle Funktionen stehen jederzeit zur Verfügung und können vom Anwender gemäß der jeweiligen Spezifikation benutzt werden. Anderenfalls bleiben die Funktionen und die zugehörigen Ein- und Ausgänge unbenutzt.

Es besteht die Möglichkeit, unterschiedliche Konfigurationen zu entwickeln um sozusagen verschiedene Modelle von Steuerungen abzubilden. Somit könnten ganze Baugruppen von Anlagen oder mechatronischen Systemen als Konfigurationen bereitgestellt und so immer wieder verwendet werden. Diese Vorgehensweise ist daher an das Phasenmodell von Kallenbach [31] angelehnt. Das Phasenmodell ist grob vereinfacht in Abbildung 4.1 dargestellt. Die erprobten Konfigurationen werden dabei in einer Bibliothek zur Verfügung gestellt und können somit schnell und einfach wiederverwendet werden.

Auch bei der herkömmlichen strukturierten Programmentwicklung werden Bibliotheksfunktionen miteinander verknüpft um damit einen Programmablauf zu generieren. Für die Entwicklung der Programme und für den Download auf die Steuerung werden die Tools der Hardwarehersteller benutzt. Bei dem hier vorgestellten fixen Funktionsmix werden allerdings keine Bibliotheksfunktionen verknüpft, sondern lediglich die fertigen

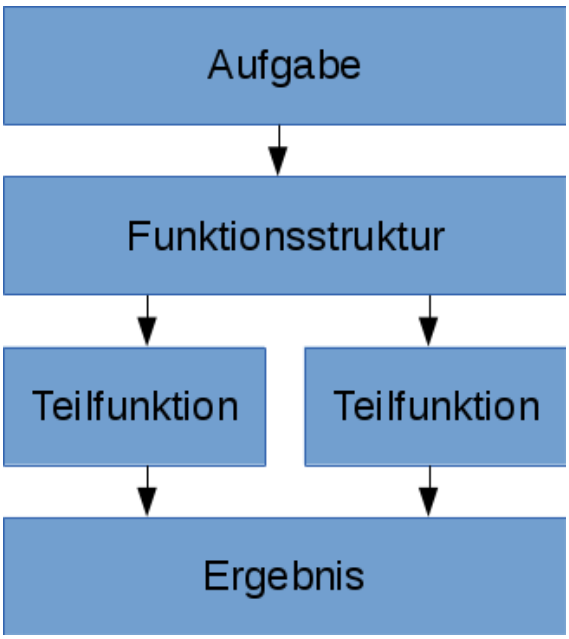

Abbildung 4.1: Phasenmodell mechatronischer Systeme nach Kallenbach [2]

Konfigurationen auf die Steuerung geladen. Die verschiedenen Konfigurationen selbst werden von Entwicklern erstellt. Dem Anwender ist es nicht möglich die Konfigurationen zu parametrieren oder zu verändern. Allerdings muss der Anwender Kenntnisse über die Werkzeuge zum Download, der fertigen Konfigurationen, auf das jeweilige Zielsystem besitzen.

4.2 Kombinierbare Funktionsblöcke

Unter kombinierbaren Funktionsblöcken soll das Zusammensetzen parametrierbarer Funktionalitäten auf einer Steuerung verstanden werden. Hierbei können definierte Funktionalitäten aus dem Leitsystem in der SPS aufgerufen und verknüpft werden. Diese Funktionalitäten müssen im SPS-Programm bereits vorhanden sein. Somit ist für die Steuerungen ein Laufzeitsystem in Form eines speziellen Programmes notwendig. Eine Sammlung der abrufbaren Funktionalitäten wird auf der Steuerung in einer Art Bibliothek organisiert. Diese Bibliothek ist mit SPS-Kenntnissen erweiterbar und muss für jeden Steuerungstyp von einem Entwickler bereitgestellt werden. Aus der Bibliothek werden Funktionen von der Leitsystemebene ausgewählt und parametriert. Die ausgewählten Funktionen werden dann von der Steuerung zyklisch bearbeitet und die Parameter an das Leitsystem zurückgegeben.

Eine sehr einfache Stackmaschine (Abschnitt 4.4) ist für Programmablauf zuständig. Im Unterschied zum fixen Funktionsmix (Abschnitt 4.1) können hier Funktionen miteinander kombiniert werden. Außerdem ist das Programm der Steuerung nicht fix vorgegeben, sondern wird durch die gewählten Funktionen und deren Verknüpfungen generiert. So-

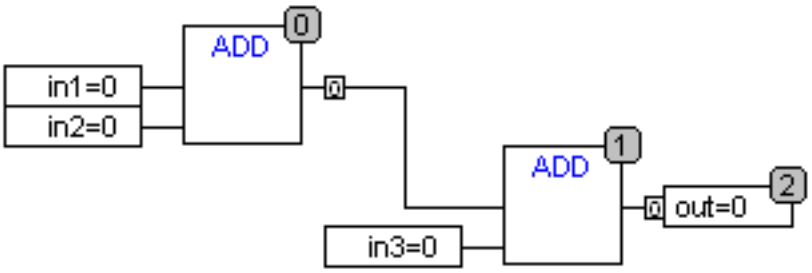

Abbildung 4.2: Continuous Function Chart Editor (CFC) [5]

mit ist es möglich, auf jeder Steuerung dieselbe Laufzeitumgebung zu verwenden und trotzdem unterschiedlichste Funktionalitäten zu realisieren. Das Laufzeitsystem muss dabei initial auf den Steuerungen in Betrieb gesetzt werden. Im Betrieb sind keine weiteren Anpassungen am SPS-Programm und somit keine Programm-Downloads notwendig. Nur zur Bereitstellung neuer Funktionalitäten in der SPS-internen Bibliothek muss das Steuerungsprogramm geändert werden. Zur Programmerstellung können die im Laufzeitsystem vorhandenen Funktionen aus einem dialoggeführten Menü ausgewählt und parametriert werden.

4.3 Freie grafische Programmierung

In Anlehnung an CFC [12] können sowohl triviale als auch komplexe Funktionsblöcke grafisch miteinander verknüpft werden. Ein einfaches Beispiel zur grafischen Programmerstellung in CFC findet sich in Abbildung 4.2. Die grafisch angeordneten Funktionsblöcke werden sequentiell abgearbeitet und bilden so das Programm. Diese Art der Programmentwicklung zeichnet sich besonders durch Einfachheit und Übersichtlichkeit aus. Anders als bei den kombinierbaren Funktionsblöcken (Abschnitt 4.2) wird hier die menübasierte Benutzerführung durch eine freigrafische Oberfläche ersetzt. Der Benutzer kann die Funktionsblöcke per Drag&Drop positionieren und per Mausklick parametrieren. Die Vernetzung erfolgt durch ziehen von Verbindungslinien zwischen den Objekten mit der Maus. Für den Betrieb der grafisch erstellten Programme ist ebenfalls ein flexibles Laufzeitsystem auf den Steuerungen notwendig.

4.4 Stackmaschine

Die Stackmaschine orientiert sich stark an der Programmiersprache Forth, welche in Abschnitt 2.4 näher beschrieben ist. Es werden nur die Stack-Verwaltung und elemen-

[12] *„Der Continuous Function Chart (CFC; deutsch Signalflussplan) ist eine Programmiersprache für Speicherprogrammierbare Steuerung (SPS). Obwohl sie keine der in der IEC 61131-3-Norm definierten Sprachen ist, stellt sie eine gängige Erweiterung von IEC-Programmierumgebungen dar. Ihr Hauptanwendungsgebiet liegt vor allem in der Prozessleittechnik, weil sich die dort auftretenden, komplexen Steuerungs- und Regelungsaufgaben sehr gut in CFC abbilden lassen.“* [15]

tare Funktionen im SPS-Programm implementiert und auf der Steuerung abgelegt. Die Stackmaschine, mit diesem elementaren Funktionsumfang, stellt hierbei wieder das Laufzeitsystem der Steuerung dar. Alle Funktionalitäten werden direkt in der Sprache der Stackmaschine implementiert. Als Programmiersprache kann Forth (Abschnitt 2.4) zum Einsatz kommen, da es sich hierbei um eine standardisierte Sprache handelt. Mittels Forth lassen sich beliebige Funktionen erstellen und auf der Steuerung ablegen. Diese Funktionen können dann wie bei den kombinierbaren Funktionsblöcken (Abschnitt 4.2) und der freien grafischen Programmierung (Abschnitt 4.3) zu einem Programm verknüpft werden.

Der große Vorteil der Stackmaschine besteht darin, dass sich der Funktionsumfang beliebig erweitern lässt. Außerdem ist die Programmiersprache Forth recht einfach zu implementieren. Forth eignet sich hervorragend für die grafisch Programmierung [34], womit eine Kombination von freier grafischer Programmierung (Abschnitt 4.3) mit der Stackmaschine naheliegt. Auch in Kombination mit funktionaler Programmierung wurde Forth bereits erfolgreich umgesetzt [34].

4.5 Bewertung der Varianten

Im Abschnitt 4.1 wird eine fix vorgegeben Konfiguration von Funktionalitäten auf der Steuerung vorgeschlagen. Hier ist die Schwierigkeit eine Konfiguration zu finden, welche für den Großteil der Anwendungsfälle passt. Die Aufgaben im Plansee-Meldesystem sind allerdings so unterschiedlich, dass fixe Funktionen nicht möglich sind. In diesem Fall würden sehr viele verschiedene Konfigurationen entstehen, welche jeweils separat gehandhabt werden müssen. Jede Konfiguration muss mit den Programmierwerkzeugen des jeweiligen Steuerungsherstellers geladen werden. Dies erfordert wieder spezielle Software und Kenntnisse des Systems. Durch die entstehende Vielfalt an Konfigurationen ist es sehr schwierig den Überblick zu behalten. Außerdem ist der fixe Funktionsmix zu unflexibel und somit für das Plansee-Meldesystem nicht brauchbar.

Alle weiteren Varianten zeichnen sich durch eine hohe Flexibilität aus. Dem Anwender wird die Möglichkeit geboten individuelle Funktionalitäten zu erzeugen. Somit können die Steuerungen anforderungsgerecht und flexibel parametriert und eingesetzt werden. Die Stackmaschine (Abschnitt 4.4) stellt dabei eine Erweiterung der kombinierbaren Funktionsblöcke (Abschnitt 4.2) dar. Gleichzeitig ist die Stackmaschine auch die flexibelste Lösung, da hier Erweiterungen des Funktionsumfanges ohne Anpassung des Laufzeitsystems [13] möglich sind. Sowohl mit den kombinierbaren Funktionsblöcken, als

[13] Als Laufzeitsystem wird in diesem Zusammenhang das Programm der Speicherprogrammierbaren-Steuerung bezeichnet. Dieses führt die parametrierbaren Funktionen aus. Es handelt sich um ein gewöhnliches SPS-Programm welches in der Entwicklungsumgebung des jeweiligen Steuerungsherstellers erstellt und initial, mit entsprechenden Systemwerkzeugen, auf der Steuerung in Betrieb genommen wird. Im Betrieb braucht das Laufzeitsystem selbst nicht angepasst werden, da die gewünschten Funktionalitäten direkt aus dem Leitsystem ausgewählt werden.

auch mit der Stackmaschine kann die freie grafische Programmierung (Abschnitt 4.3) kombiniert werden. Somit kann die freie grafische Programmierung jeweils als Erweiterung der kombinierbaren Funktionsblöcke und der Stackmaschine gesehen werden.

Da für die kombinierbaren Funktionsblöcke ein abgespecktes Stack-System zum Einsatz kommt, stellt die Stackmaschine dazu eine Erweiterung dar. Bei der Stackmaschine kann im Unterschied zu den kombinierbaren Funktionsblöcken der Funktionsumfang im laufenden Betrieb angepasst werden. Für die Stackmaschine wird ein Forth-System implementiert und Forth als Programmiersprache verwendet. Nähere Informationen zu Aufbau und Funktion von Forth finden sich im Abschnitt 2.4. Mit der Programmiersprache Forth lassen sich dann komplette Programme erstellen und auf dem Laufzeitsystem der Steuerung ausführen. Dabei finden die Programmerstellung, -übertragung und -inbetriebnahme direkt im Leitsystem statt. Über definierte Kommunikationskanäle zwischen Steuerung und Leitsystem kann das Programm interaktiv erstellt, bearbeitet und geladen werden.

Obwohl die Stackmaschine das langfristige Entwicklungsziel darstellt, habe ich mich in der vorliegenden Arbeit für die Varianten mit den kombinierbaren Funktionsblöcken entschieden. Diese Variante ist leicht zu implementieren und bietet eine einfache Handhabung für die Benutzer. Außerdem ist eine Weiterentwicklung in Richtung Forth jederzeit möglich und kann inkrementell durchgeführt werden. Es werden definierte Funktionsblöcke für Beckhoff-Steuerungen entwickelt. Diese können dann über einen Assistenten im Leitsystem ausgewählt und parametriert werden. Die so erstellte Konfiguration wird dann aus dem Leitsystem auf die Steuerung übertragen und dort ausgeführt. Außerdem werden für ausgewählte Funktionen auch Leitsystem-Scripts erstellt um die Ausführung der Funktionalität auch direkt im Leitsystem zu ermöglichen.

Mit den kombinierbaren Funktionsblöcken kann so die Flexibilität und Zuverlässigkeit des Laufzeitsystems getestet werden. Außerdem können Erfahrungen über die Verwendung und Handhabung durch die Benutzer gesammelt werden. Da das Laufzeitsystem für die Funktionsblöcke einfacher aufgebaut ist als die Stackmaschine, kann auch mit schnellen Ergebnissen gerechnet werden. Ebenfalls als zukünftige Erweiterung wird die freie grafische Programmierung weiterverfolgt. Da diese sowohl für die Funktionsblöcke als auch für die Stackmaschine verwendbar ist, kann diese rasch getestet werden. Das Laufzeitsystem und die Funktionsblöcke werden im Rahmen dieser Arbeit beispielhaft für ein konkretes Problem entwickelt und beschrieben. Dabei wird die Steuerung und Überwachung der Notstromaggregate am Standort Reutte in das Plansee-Meldesystem integriert. Zur konkreten Implementierung siehe Abschnitt 3.4.

5 Systementwicklung

Die Systementwicklung erfolgt bei Plansee nach einer definierten Vorgehensweise. Bei der Entwicklung werden aus den Spezifikationen zuerst Testfälle abgeleitet und zugehörige Testroutinen entwickelt. Anschließend werden die Funktionalitäten implementiert und im Testsystem getestet. Für die Entwicklung des Laufzeitsystems für Softwareobjekte wird im Plansee-Meldesystem die Leitsystemsoftware WinCC OA von Siemens eingesetzt. Als Speicherprogrammierbare-Steuerungen, sogenannte dezentrale Knoten, werden Steuerungen der Firma Beckhoff eingesetzt. Eine detaillierte Beschreibung der eingesetzten Hard- und Software findet sich in Kapitel 2.6.

Im Zuge dieser Diplomarbeit wird das Laufzeitsystem für Softwareobjekte auf einer Beckhoff CX8090 Steuerung entwickelt. Zusätzlich werden einige Funktionalitäten und Bausteine erstellt, welche in diesem Laufzeitsystem ablaufen können. Als konkrete Anwendung soll die Steuerung der Notstrom-Diesel-Aggregate implementiert und die Funktionalität und Handhabung gezeigt werden. Dazu werden sowohl die Softwarekomponenten für die Beckhoff-Steuerung als auch die Parametrier- und Diagnosefunktionen im Leitsystem entwickelt. Um die Weiterentwicklung des Systems und die Bereitstellung weiterer Funktionalitäten durch andere Entwickler zu vereinfachen, wird eine umfangreiche Dokumentation mit Beispielen und Systembeschreibungen im Plansee-Wiki-System [14] bereitgestellt.

Somit stellt das Entwicklungsmodell mit den Werkzeugen GIT und Redmine und den festgelegten Code-Richtlinien die Basis für jede Entwicklung im Plansee-Meldesystem dar. In den folgenden Abschnitten werden diese Vorgehensweisen näher erläutert.

5.1 Entwicklungsmodell

Für die Software-Entwicklung werden Git und Redmine eingesetzt. So ist es möglich jedem Entwickler eine standardisierte Vorgehensweise zur Code-Entwicklung und für die Kollaboration zu bieten. Um eine reibungslose Zusammenarbeit zu gewährleisten, müssen einige Regeln festgelegt werden, an die sich alle Projektbeteiligen zu halten

[14] *„Ein Wiki (hawaiisch für „schnell“), seltener auch WikiWiki oder WikiWeb genannt, ist ein Hypertextsystem für Webseiten, deren Inhalte von den Benutzern nicht nur gelesen, sondern auch online direkt im Webbrowser geändert werden können (Web-2.0-Anwendung). Das Ziel ist häufig, Erfahrung und Wissen gemeinschaftlich zu sammeln (kollektive Intelligenz) und in für die Zielgruppe verständlicher Form zu dokumentieren. Die Autoren erarbeiten hierzu gemeinschaftlich Texte, die ggf. durch Fotos oder andere Medien ergänzt werden (Kollaboratives Schreiben, E-Collaboration). Ermöglicht wird dies durch ein vereinfachtes Content-Management-System, die sogenannte Wiki-Software oder Wiki-Engine. Die bekannteste Anwendung von Wikis ist die Online-Enzyklopädie Wikipedia, welche die Wiki-Software MediaWiki einsetzt. Zudem nutzen auch viele Unternehmen Wikis als Teil von dem Wissensmanagementsystem in ihrem Intranet (standortübergreifend), siehe Enterprise Wiki.“* [21]

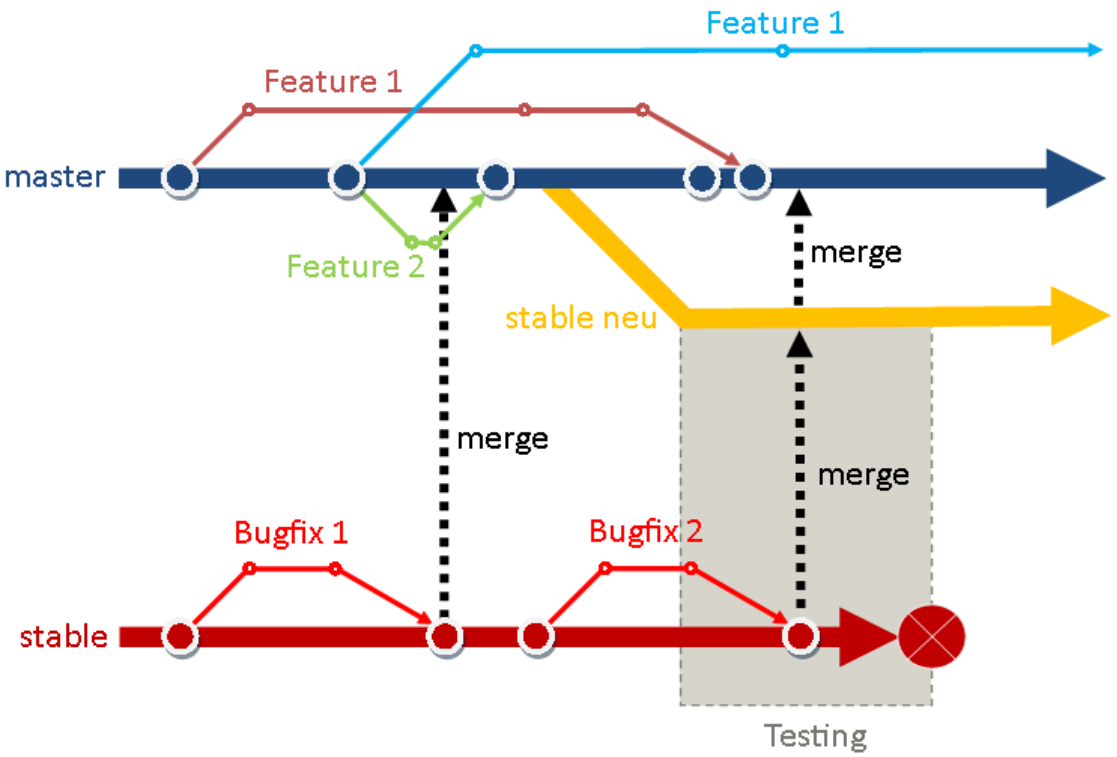

Abbildung 5.1: Git Branching-Modell

haben. Dieses Regelwerk sorgt für gleichbleibend hohe Code-Qualität und sichert Verwendbarkeit und Dokumentation aller Software-Änderungen.

5.1.1 GIT

Git ist ein Werkzeug mit dem eine Versionsverwaltung realisiert werden kann. Die Versionsverwaltung bringt den Vorteil, dass mehrere Entwickler gleichzeitig an der gleichen Codebasis arbeiten können. Es kann auf die komplette Historie aller Daten zugegriffen werden und Änderungen können einfach rückgängig gemacht werden. Git unterstützt dabei zentrale und dezentrale Entwicklungsmodelle, wobei wir uns auf das zentrale Modell beschränken werden. Beim zentralen Entwicklungsmodell gibt es ein zentrales Repository, indem alle Änderungen von den Entwicklern gesammelt werden. Jeder Entwickler kann von diesem Repository die neusten Versionen beziehen. Git unterstützt das Arbeiten mit mehreren parallelen Entwicklungszweigen. So können einfach neue Zweige, von beliebigen vorhandenen Zweigen, abgespalten werden. Wenn die neuen Codeteile wieder in den Hauptzweig integriert werden sollen, können diese Zweige einfach wieder zusammengeführt werden. Jeder Entwickler hat die Möglichkeit beliebig zwischen den einzelnen Zweigen hin und her zu wechseln. Soll beispielsweise im stabilen Zweig ein Fehler ausgebessert werden, so kann einfach ein Checkout von diesem Zweig gemacht werden. Nachdem der Fehler behoben ist, kann der Entwickler seine vorherige Arbeit aus einem anderen Zweig wieder aufnehmen.

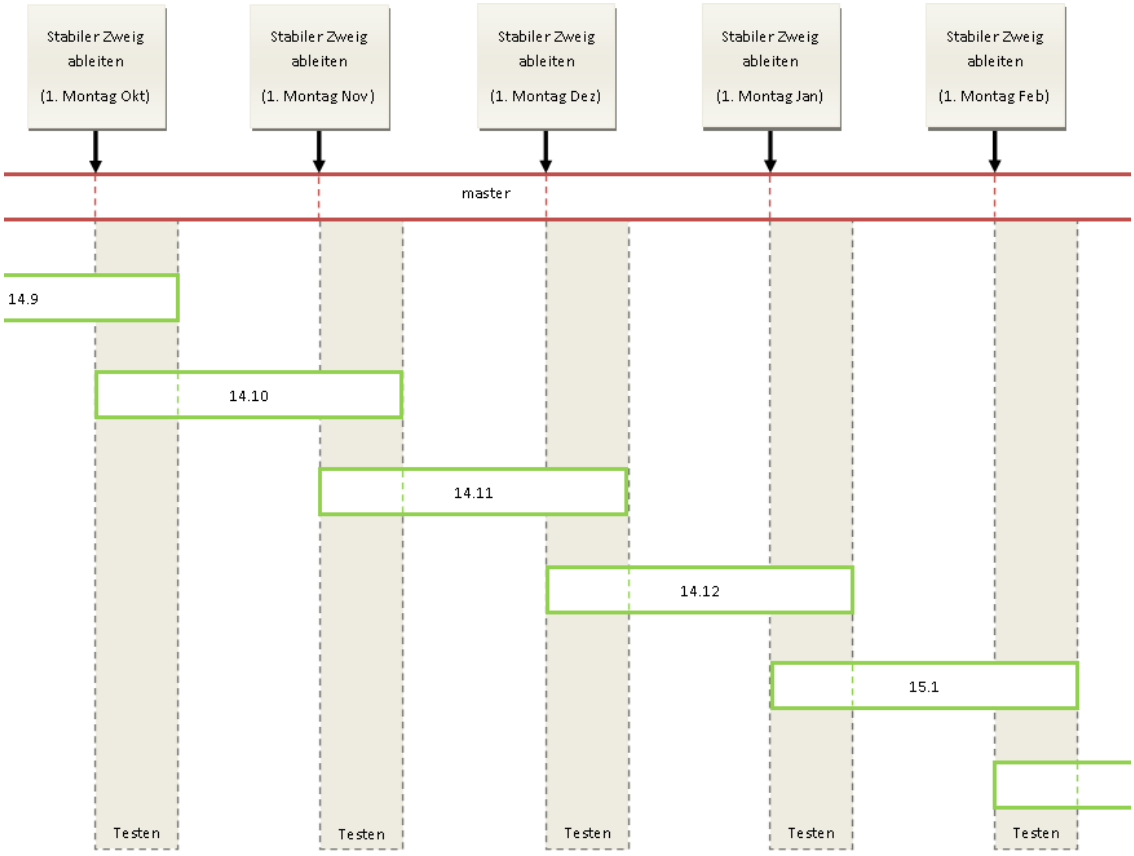

Abbildung 5.2: Git release Plan

Im Master Zweig werden neue Features entwickelt. Jedoch sollten die Programmierer nicht direkt in Master arbeiten, sondern davon immer neue Feature-Zweige ableiten (siehe Bild 5.1) und dort auch alle Änderungen durchführen. Wenn die Features in den jeweiligen Zweigen soweit gereift und stabilisiert sind, dass diese einsatzbereit sind, dann soll einen Merge Request (bzw. Pull-Request) von diesem Zweig in den Master Zweig geöffnet werden. Bei diesem Merge Request haben andere Programmierer dann die Aufgabe einen Code-Review durchzuführen. Wenn die Codequalität das nötige Niveau erreicht hat, darf das Feature in den Master-Zweig wandern. Dieser Workflow ermöglicht es den Master-Zweig immer in einem releasebaren Zustand zu halten. Somit ist es möglich, in fix definierten Abständen daraus stabile Zweige abzuleiten. Außerdem können Programmierer mit dieser Arbeitsweise neue Features immer auf der Basis von funktionierendem Code aufbauen. Fehlerkorrekturen sind in Master immer erlaubt, jedoch ist auch hier ein Review-Prozess sinnvoll. Diese Arbeitsweise wird auch „Topic Branches“ genannt [9].

In den stabilen Zweigen dürfen nur Fehler ausgebessert werden. Neue Features sind hier auf keinen Fall erlaubt. Fehlerverbesserungen an Kernbestandteilen sollten gut überlegt werden und nur gegebenenfalls im Master durchgeführt werden. Alle Patches für stabile Zweige, sollten auch den Review-Prozess durchlaufen. Die Fehlerkorrekturen der stabilen Zweige werden immer wieder in den Master Zweig gemerged. Diese

Arbeitsweise wird auch „Long-Running Branches“ genannt [9].

In gewissen Abständen wird vom Master-Zweig ein neuer stabiler Zweig abgespalten. Dabei werden alle neuen Funktionen, die bisher den Master-Zweig erreicht haben, in das Produktivsystem übernommen. Dieser Zweig wird dann eine gewisse Zeit lang getestet und stabilisiert bis er schlussendlich stabil genug für das Produktivsystem ist. Während der Stabilisierung des neusten stabilen Zweiges wird zusätzlich noch der momentan verwendete stabile Zweig mitgepflegt. Änderungen im alten stabilen Zweig müssen in den neuen stabilen Zweig und in den Master Zweig gemerged werden. Nach Ablauf der Testphase wird der alte stabile Zweig aufgegeben und nur noch der neue stabile Zweig gepflegt.

Jeden ersten Montag vom Monat wird ein neuer stabiler Zweig vom Master-Zweig abgespalten. Der neue Zweig enthält alle bis dahin fertigen Features. Dieser neue Zweig wird dann eine Woche lang getestet und stabilisiert, bis er schlussendlich den aktuellen stabilen Zweig vom Produktivsystem ersetzt. Stabile Zweige werden wöchentlich ins Produktiv-System übertragen. Bei sehr dringenden Fehlerkorrekturen kann der stabile Zweig auch sofort übernommen werden.

Durch gut gewählte Release-Abstände ist die Anzahl an neuen Features gut überschaubar und somit können neue Codeteile besser getestet werden. Korrekturen können schneller durchgeführt werden, da die Zeit zwischen Fertigstellung von einem Feature und Stabilisierung von einem neuen stabilen Zweig nicht all zu groß ist. Dies verringert die benötigte Zeit zum Wiedereinlesen in den Code. Features kommen nur in das Release wenn sie wirklich fertig sind. Abnehmende Codequalität, kurz vor dem Release, sollte damit vermieden werden, da der Programmierer sich nicht auf die ToDo-Liste fixieren muss. Wenn es sich für ein bestimmtes Release nicht ganz ausgeht, dann kommt ein Feature eben mit 4 Wochen Zeitverzögerung ins Produktivsystem. Dadurch sind auch Testphasen besser planbar, da Releases in fixen Abständen stattfinden. Diese Vorgehensweise wird in Bild 5.2 dargestellt.

5.1.2 Redmine

Redmine ist eine flexible Webapplikation zur Projektverwaltung und Aufgabenverteilung. Es kann als Ticketsystem verwendet werden und dient der Organisation von Teams. Aufgaben werden dabei in Form von Tickets organisiert. Als Projektmanagement-Methode nutzen wir „Kanban“. Bei dieser Methode werden Aufgaben für die kommende Woche von den Mitarbeitern übernommen und selbständig abgearbeitet. Jeder Mitarbeiter hat somit die Möglichkeit sich seine Aufgaben aus dem Backlog zu holen und zu bearbeiten. Die aktuellen Arbeiten aller Projektmitarbeiter sind am Task-Board zu sehen. Während der Arbeit an einer Aufgabe werden sämtliche Informationen im zugehörigen Ticket dokumentiert. So können alle Mitarbeiter den Status der Aufgabe verfolgen und

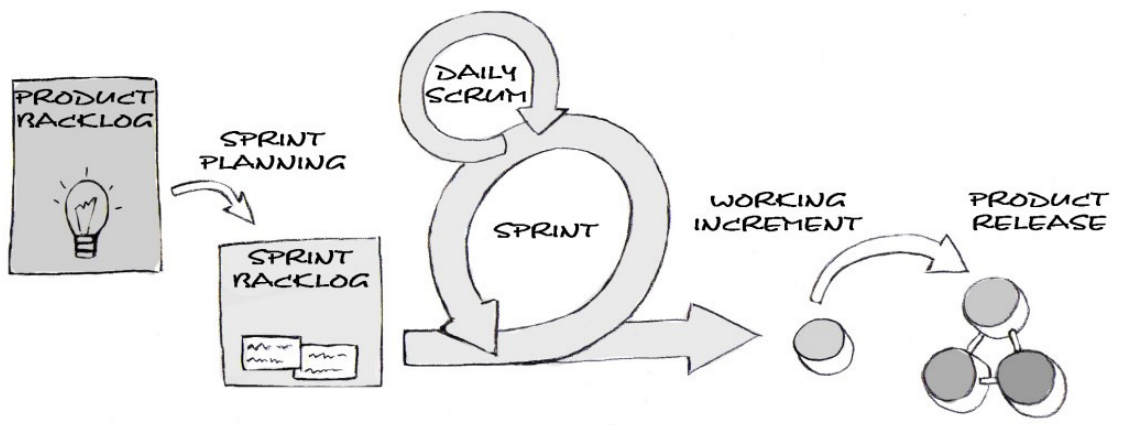

Abbildung 5.3: Kanban-Modell [32]

weiterführende Informationen dazu abrufen. Diese Vorgehensweise ist in Bild 5.3 dargestellt.

Redmine bietet zusätzlich ein Wiki-System welches für Dokumentationen genutzt wird. Alle Dokumentationen zu den Projekten sowie Anleitungen und sonstige Dokumente werden im Wiki abgelegt. Alle Wiki-Seiten sind mit einem Webbrowser aufruf- und änderbar. Jede Seite besitzt eine Historie, mit der alle Änderungen verfolgt werden können. Das Redmine-Wiki stellt somit die zentrale Dokumentenverwaltung für alle Themen zum Plansee-Meldesystem dar.

Auch die GIT-Repositories sind in Redmine integriert. So können Changesets direkt zu den entsprechenden Tickets angehängt werden. Auch Merge-Requests zur Code-Review werden mit Redmine abgebildet. Damit kann nicht nur der Source-Code mehrerer Entwickler verwaltet sondern auch die Softwareentwicklung zentral koordiniert werden. Fehler werden in Form von Bugreports von den Benutzern im Redmine-System gemeldet und können so umgehend und transparent bearbeitet werden.

Egal ob ein einzelner Entwickler oder ein ganzes Team an einer Aufgabe arbeitet. Mit dem Regelwerk zur Aufgabenbearbeitung und Dokumentation mit Redmine kann eine durchgängige Systementwicklung gewährleistet werden. Benutzer und Entwickler können sich ständig über Fortschritte und Änderungen am System informieren und aktiv die Entwicklung mitgestalten. Wichtige Erkenntnisse und andere Informationen können von allen Benutzern im Wiki-System dokumentiert werden. Bei Problemen und Unklarheiten können andere Benutzer diese Dokumente abrufen und von den Erfahrungen anderer Nutzer profitieren.

Abbildung 5.4: Aufbau eines Software-Objektes für das Laufzeitsystem

5.2 Steuerungskonzept

Das Steuerungssystem basiert auf Objekten welche in einem Stack verwaltet werden. Dabei wurden einige Merkmale eines Forth-Systems, aus Abschnitt 2.4, übernommen. Alle Objekte welche im Laufzeitsystem ablaufen sind in einem Stack-Speicher abgelegt. Dieser Speicherbereich enthält Zeiger auf die Datenstrukturen der jeweiligen Objekte. In der Datenstruktur eines Objektes ist der gesamte Zustand des Objektes gespeichert. Diese Beziehung ist in Abbildung 5.5 dargestellt. Abbildung 5.4 zeigt die Datenstruktur eines Objektes. Jedes Objekt hat folgenden Aufbau:

- Objekt-ID: Eindeutige Identifizierung des Objektes.
- Objekt-Typ: Der Typ des Objektes verweist auf den Laufzeitcode. Dieser wird durch den Objekt-Stack aufgerufen und enthält den jeweiligen Programmcode.
- Objekt-Einstellungen: In den Einstellungen wird die Konfiguration des Objektes gespeichert. Diese werden vom Leitsystem parametriert.
- Objekt-Daten: Der Datenbereich enthält interne Daten des Objektes.
- Objekt-Werte: Hier werden Daten vom Leitsystem abgelegt welche an das Objekt übergeben werden. Die Daten können vom Objekt manipuliert und an das Leitsystem zurückgegeben werden.

Wenn das Laufzeitsystem gestartet ist, wird der Objekt-Stack zyklisch abgearbeitet. Dabei werden alle Objekte nach der Reihe durchlaufen und deren Laufzeitcode ausgeführt. Während der Ausführung werden Daten aus dem Speicherbereiche des Objektes eingelesen, verarbeitet und ausgegeben. Für die Adressierung der Objekt-Datenstruktur während der Abarbeitung werden diverse Zeiger verwendet. Diese Zeiger werden jeweils am Beginn der Bearbeitung des Objektes initialisiert und dienen zur Orientierung innerhalb der Objekt-Daten.

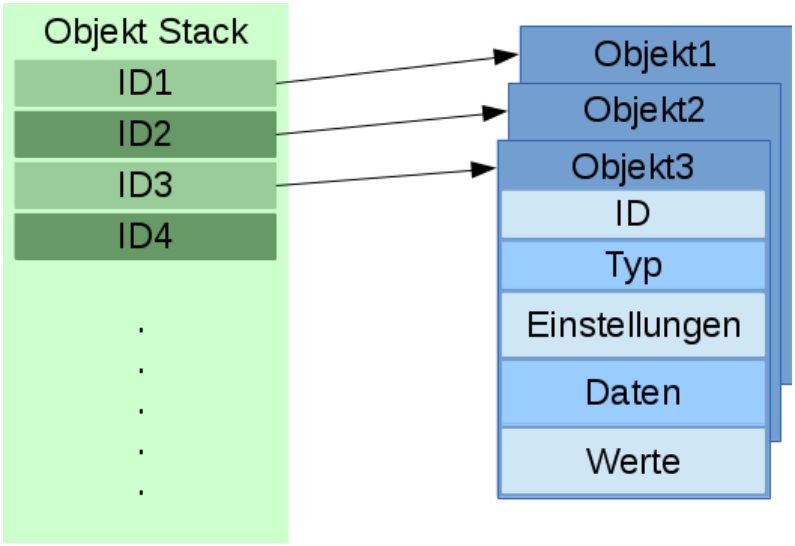

Abbildung 5.5: Modell des Objeckt-Stack

Der Laufzeitcode jedes Objektes wird mit den Werkzeugen des Steuerungs-Herstellers, in unserem Fall Beckhoff, erstellt und auf die Steuerung geladen. Somit ist dieser Teil des Laufzeitsystems. Im Unterschied zu Forth können Erweiterungen der Funktionalität nicht im System selbst generiert werden. Jedes neue Objekt muss im Code des Laufzeitsystems implementiert werden. Zwar gibt es Objekte welche die Verknüpfung zwischen Objekten ermöglichen, diese eignen sich aber nur begrenzt zur Generierung gänzlich neuer Funktionalitäten. Um die Erweiterung des Laufzeitsystems so einfach wie möglich zu gestalten, wurde eine umfangreiche Bibliothek, mit diversen Funktionen zur Objektgenerierung, erstellt. Außerdem gibt es eine Auswahl an Beispielcodes und Richtlinien zum Design und zur Implementierung neuer Objekte. Durch die Verzeigerung der Programmausführung sind nicht alle Funktionalitäten, welche die Software des Steuerungssystems normalerweise bietet, zulässig. Besondere Vorsicht ist bei Verzweigungen oder Instanziierungen geboten. Da hier der Programmfluss verändert wird, sind spezielle Funktionen für die Rückkehr in den Programmablauf des Laufzeitsystems notwendig. Bei der Erstellung von Tests für das Laufzeitsystem und bei diversen Implementierungen hat sich allerdings gezeigt, dass die Entwicklung neuer Objekte recht einfach und schnell von statten gehen kann. Für eine reibungslose Objekt-Entwicklung sind Übung, eine strukturierte Vorgehensweise, sowie die Kenntnis der Richtlinien wichtig.

5.3 Stack-System

Ein Stack, zu Deutsch Stapel, beschreibt eine Datenstruktur im Speicher welche nach dem last-in-first-out (LIFO) Prinzip funktioniert. Das bedeutet, dass Elemente nur oben auf dem Stack dazu gefügt oder entnommen werden können [6]. Auch für das Laufzeitsystem stellen Stapelspeicher eine sehr einfache und effiziente Möglichkeit der Da-

tenorganisation dar. Stacks wurden deshalb gewählt, weil sie dynamisch wachsen und schrumpfen können und dabei nahezu keine Verwaltung benötigen. In der vorliegenden Implementierung werden zwei Stapelzeiger verwendet, wobei ein Zeiger auf das aktuelle Element und der Andere auf die oberste Stack position zeigt.

Eigentlich besteht das Stack-System aus zwei Stacks. Im Gegensatz zu Forth (Abschnitt 2.4) handelt es sich dabei aber nicht um den Daten- und den Rückgabe-Stack. Das Laufzeitsystem für die Software-Objekte besteht aus einem Objekt-Stack welcher alle vorhandenen Objekte enthält und zyklisch abgearbeitet wird. Auch die Objekt-Datenstrukturen werden als Stapel im Speicher verwaltet. Für jeden Stack gibt es die oben beschriebenen Zeiger. Hier besteht das Problem, dass die Datenstrukturen der Objekte keine fixen Länge haben. Dieser Umstand stellt zwar bei der sequentiellen Abarbeitung der Objekte kein Problem dar, macht sich aber beim Ändern oder Löschen von Objekten bemerkbar. Um keine Lücken im Speicher zu erhalten, wird daher bei Änderungen im Objekt-Stack eine Reorganisation des Objektdaten-Stacks durchgeführt. Die Implementierung der Datenstrukturen für die Objekte ließe sich mit einer verketteten Liste besser lösen. Eine Änderung dieser fundamentalen Designschwäche war in der vorliegenden Version des Laufzeitsystems nicht mehr möglich. Bei zukünftigen Versionen wird dieses Problem jedoch behoben werden.

5.4 Kommunikation und Datenaustausch

Für den Datenaustausch zwischen Steuerung und Leitsystem werden zwei grundsätzliche Verfahren unterschieden. Wie im Abschnitt 5.2 beschrieben, wird zwischen Objekt-Einstellungen und Objekt-Werten unterschieden. Zum Einen findet eine zyklische Aktualisierung der Werte eines Objektes statt, zum Anderen werden die Einstellungen der Objekte nur bei Bedarf vom Leitsystem zu der Steuerung übermittelt.

5.4.1 Austausch der Objekt-Werte

Für den Zugriff über Modbus-TCP werden die Objekt-Datenstrukturen in einem definierten Merkerbereich in der Steuerung abgelegt. Dadurch ist eine eindeutige Adressierung der jeweiligen Daten möglich. Die Werte der Objekte werden dabei direkt vom Leitsystem gelesen oder geschrieben. Das Auslesen der Werte erfolgt zyklisch über einen definierten Modbus-Poll-Intervall. Schreibzugriffe aus dem Leitsystem an die Steuerung werden ereignisgesteuert ausgelöst. Um den direkten Zugriff auf die Daten zu ermöglichen benötigt das Leitsystem Informationen über die jeweiligen Objekte. Dazu werden Objekt-Kataloge im Leitsystem angelegt, welche die benötigten Informationen enthalten. Die Informationen zu den Objekten sind dabei nicht nur für die Kommunikation sondern auch für die Objektlaufzeit im Leitsystem selbst notwendig. Werden neue Objekttypen in der Laufzeitumgebung der Steuerung implementiert, müssen diese auch im

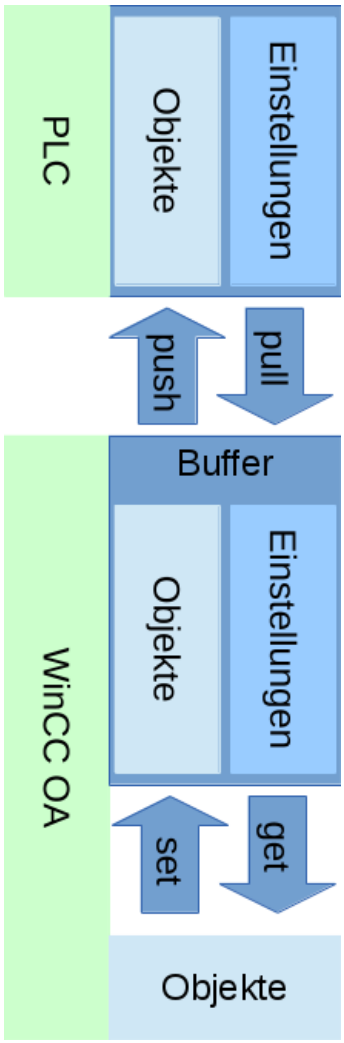

Abbildung 5.6: Modell des Datenaustausches zwischen Steuerung und Leitsystem

Leitsystem angelegt werden. Ansonsten ist der Zugriff auf die Objekte des Typs nicht möglich.

5.4.2 Zugriff auf die Objekt-Parametrierung

Für die Objekt-Parametrierung wird ein Abbild der Objektdaten auf der Steuerung auch im Leitsystem erzeugt. Die Bereiche für die Objekt-Werte werden dabei ausgeblendet, da diese direkt adressiert werden. Zum Einlesen und Schreiben der Daten wurden die Kommunikationsprozeduren PULL und PUSH definiert. PULL liest die Daten eines definierten Objektes aus der Steuerung aus und legt diese in einem Puffer im Leitsystem ab. PUSH transferiert den Bufferinhalt vom Leitsystem in die Steuerung. Wird eine Objekt-Datenstruktur mit einer nicht vorhandenen ID auf die Steuerung geschrieben, wird das gewünschte Objekt angelegt. Zum Ändern eines Objektes auf der Steuerung werden spezielle Flags im Datenbereich verwendet.

Abbildung 5.7: Auswahl eines definierten Objektes aus dem Katalog

Mit den Befehlen SET und GET wird der Objekt-Puffer innerhalb des Leitsystems manipuliert. Um Änderungen an einem bestimmten Objekt durchzuführen, werden die Daten des Objektes zuerst in den Leitsystem-Puffer eingelesen. Anschließend werden die Änderungen im lokalen Puffer durchgeführt und wieder auf die Steuerung geschrieben. Um Inkonsistenzen durch gleichzeitige Änderungen von mehreren Leitsystem-Stationen zu verhindern wurde ein System zur gegenseitigen Verriegelung implementiert. Dadurch ist sichergestellt, dass Einstellungen nur von jenem System gesetzt werden können welches die Daten zuletzt gelesen hat. Erst wenn dieses System die Steuerung wieder freigibt, können andere Systeme Änderungen darauf durchführen.

Eine schematische Darstellung der Abläufe bei der Parametrierung von Objekten ist in Abbildung 5.6 zu finden.

5.5 Programmerstellung

Grundsätzlich ist das System der Laufzeit-Objekte nicht für die komplexer Steuerungs- oder Regelungsaufgaben gedacht. Ein Programm besteht daher meist aus wenigen, meist unabhängigen Objekten. Dennoch besteht die Möglichkeit Objekte mit speziellen Datenaustausch-Funktionen zu koppeln. Da für die Programmerstellung nur einfache Parametrierdialoge vorhanden sind, verliert der Benutzer meist schon bei wenigen Verknüpfungen den Überblick. Allerdings sind die Dialoge zur Erstellung einfacher Steuerungsaufgaben sehr gut geeignet. In Abbildung 5.7 wird ein Dialog zur Auswahl gewünschter Objekte aus dem Katalog dargestellt.

Zur Auswahl der Funktionalitäten auf der Steuerung werden die gewünschten Objekte aus einer Liste von Objekttypen ausgewählt. Je nach Objekttyp können über die Eingabemaske diverse Funktionen und Einstellungen ausgewählt werden. Wenn notwendig, können Ein- und Ausgangsadressen auf der Steuerung mit dem Objekt verknüpft wer-

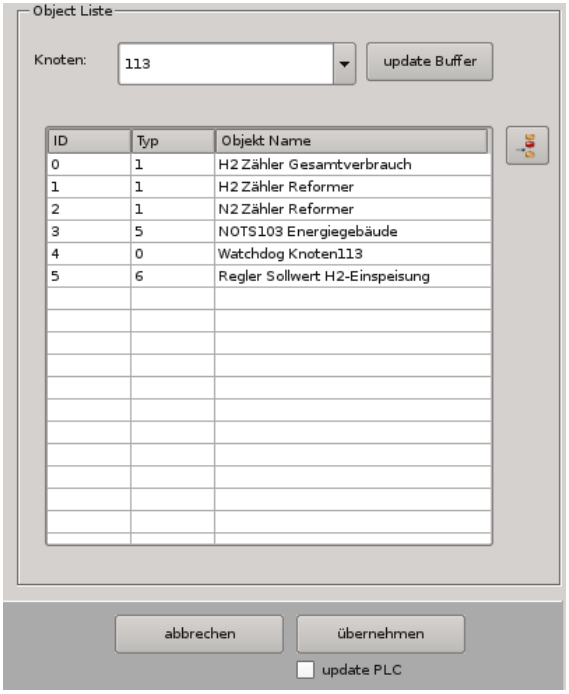

Abbildung 5.8: Ansicht aller Objekte auf der Steuerung

den. Nach der Parametrierung des Objekts wird dieses automatisch in der Objekttabelle eingefügt und der Objekt-Puffer gesetzt. Das System berechnet anhand der Objekt-Id alle Modbus-Adressen und verknüpft diese mit den Objekt-Werten im System. Der Benutzer hat nun die Möglichkeit weiter Objekte hinzuzufügen, zu bearbeiten oder zu entfernen. Anschließend kann die gewählte Konfiguration auf die Steuerung geschrieben werden. Eine Objekttabelle ist in Abbildung 5.8 dargestellt.

Wenn alle Einstellungen korrekt auf die Steuerung geladen wurden, beginnt das Laufzeitsystem sofort mit der zyklischen Bearbeitung aller Objekte in der Objekt-Tabelle. Die Werte werden dann umgehen im Leitsystem angezeigt. Eventuelle Fehler oder Warnungen bei der Abarbeitung der Objekte im Laufzeitsystem erscheinen im Log-Viewer (siehe Abschnitt 6.3).

Durch diese einfache Methode der funktionalen Programmerstellung sind auch Benutzer ohne Erfahrung mit Speicherprogrammierbaren-Steuerungen in der Lage schnell Funktionalitäten auf den Steuerungen im Feld zu implementieren. Die enge Führung über die Eingabemasken verhindert Fehler und ermöglicht die einfache Parametrierung der Funktionen. In vielen Fällen macht allerdings die Verknüpfung der Objekte mit den Ein- und Ausgängen der Steuerungen Schwierigkeiten. Hier muss der Benutzer direkt die Adressen der Peripherie-Baugruppen angeben. Viele Benutzer wissen aber zu wenig über den Aufbau der Steuerungen bescheid und finden sich nicht zurecht. Hier sollte noch eine grafische Ansicht für die Auswahl der Adressen der Ein- und Ausgänge erstellt werden. In dieser Ansicht sollen die parametrierten Signale der Ein- und Ausgänge angezeigt werden und grafisch auswählbar sein.

6 Integration und Tests

Eine Integration in den Bestand kommt nur im laufenden Betrieb in Frage. Während der Umstellung müssen alle Systeme in vollem Umfang aktiv sein. Um eine reibungslose Umstellung durchführen zu können, müssen alle Komponenten zur Inbetriebnahme problemlos funktionieren. Dazu werden Entwicklung und Tests in einem unabhängigen Entwicklungssystem durchgeführt. Diese Entwicklungssystem kann außerdem, wie in Abschnitt 6.4 beschrieben, für die Schulungen der Benutzer verwendet werden. Um einen reibungslosen Testablauf zu gewährleisten wurde das Testframework, welches schon im Praxisprojekt „Verteilte Feldebene für das Plansee-Meldesystem“ [23] verwendet wurde, weiter entwickelt. Für die Qualitätssicherung der erstellten Funktionalitäten werden laufende Tests an einem Testserver durchgeführt. Auch nach der Systemumstellung werden die Tests weiterhin beibehalten. So kann nur getesteter Code in das Produktivsystem gelangen.

Vor der Integration des Laufzeitsystems in das Produktivsystem wurde die Vorgehensweise der Umstellung mit allen beteiligten Modulen am Testsystem durchgespielt. So konnten einige Probleme bereits vor der Inbetriebnahme umgangen werden. Die gesamte Inbetriebnahme konnte dadurch komplett in den Betriebsferien über Weihnachten durchgeführt werden.

6.1 Tests

Ein Kernelement des neu etablierten Entwicklungsmodelles bei Plansee sind umfangreiche Tests. Die Tests dienen vor allem der Absicherung der Systeme bei Änderungen. Außerdem unterstützen Tests bei der Software Entwicklung, da ein umgehender Funktionsnachweis erbracht werden kann. Grundsätzlich werden die Tests vor dem Code geschrieben um die Funktionalität der Software ganzheitlich zu definieren. Somit sind Tests auch Teil der Dokumentation, da Funktionalität und Anwendung der Software einfach und klar ersichtlich sind. Es soll eine möglichst hohe Testabdeckung erreicht werden. Dazu werden bereits bei der Entwicklung des Quellcodes alle definierten Bedingungen durch Testroutinen geprüft. Bei einer Fehlerbehebung wird zuerst ein Test geschrieben der den Fehler reproduziert. Anschließend wird der Fehler im Code behoben [2] [31].

Wir unterscheiden drei Arten von Tests: [15]

- Unit-Test: Test aller Funktionen einer Modul-Bibliothek. Es werden keine Laufzeit-

[15] In der Literatur werden unterschiedlichste Tests und Testmethoden vorgeschlagen. Um Missverständnisse auszuschließen haben wir die bei Plansee verwendeten Testmethoden eigens definiert.

Abbildung 6.1: Ausschnitt aus der Wiki-Doku

Scripts oder Verbindungen zu anderen Modulen getestet.

- Integration-Test: Es werden die Bibliotheken gemeinsam mit den Laufzeit-Funktionalitäten eines Modules getestet.
- System-Test: Es wird das Zusammenspiel von Modulen getestet. Hier werden Abhängigkeiten und Wechselwirkungen sichtbar.

Natürlich sind nicht für alle Module alle Arten von Tests sinnvoll. Welche Tests für welche Software-Module angewendet werden, hängt von der jeweiligen Funktionalität ab. Beispielsweise machen Integrations-Tests bei reinen Verwaltungs-Funktionalitäten ohne Laufzeit-Code keinen Sinn. Um den Testprozess transparent und durchgängig zu realisieren wurde im Projektverwaltungssystem eine umfassende Sammlung an Beispielen, Code-Ausschnitten und Beschreibungen hinterlegt. Die durchgängige Dokumentation soll sicherstellen, dass das Testsystem zukünftig von allen Entwicklern einheitlich verwendet wird. Ein Ausschnitt aus der Dokumentation im Wiki ist in Abbildung 6.1 dargestellt.

6.2 Testsysteme

Um die Funktion des Gesamtsystems sicherzustellen sind Tests auf mehreren Ebenen notwendig.

1. Laufzeitsystem auf der Steuerung: Hier werden die Grundfunktionen, als Voraussetzung für einen Programmablauf auf der Steuerung, getestet.
2. Software-Objekte für die Steuerung: Tests der Funktionalitäten für die Steuerung

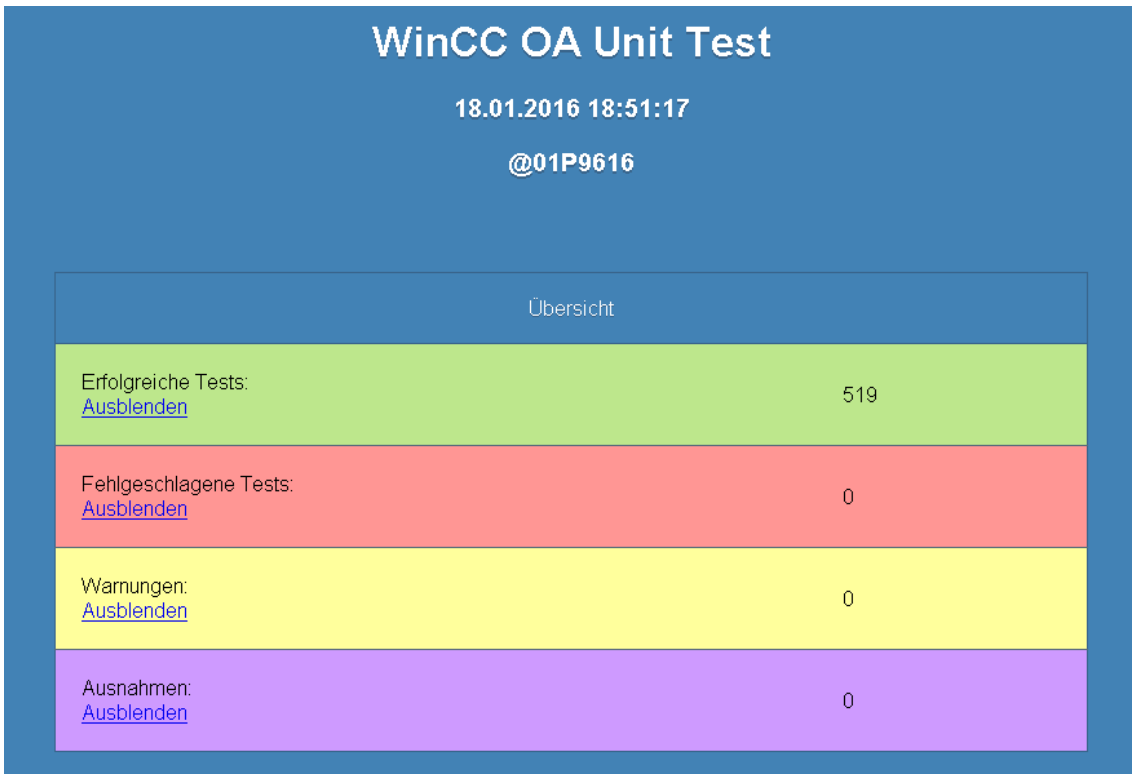

Abbildung 6.2: Rückmeldung des automatischen Testsystems

an sich.

3. Kommunikation zwischen Leitsystem und Steuerung
4. Parametrierungs- und Überwachungsebene im Leitsystem
5. Laufzeitsystem im Leitsystem: Code für den Ablauf von Software-Objekten im Leitsystem
6. Software-Objekte für das Leitsystem

Da für ein funktionierendes System alle Ebenen zusammenspielen müssen, sollen diese auch alle im Testsystem berücksichtigt werden. Es müssen sowohl Codes auf der Beckhoff-SPS als auch im Leitsystem getestet werden. Außerdem muss die ordnungsgemäße Kommunikation zwischen SPS und Leitsystem geprüft werden. Für die automatischen Tests wurde ein Testserver mit Beckhoff Twincat2 und Siemens WinCC OA aufgesetzt. Für die Kommunikation wird Modbus-TCP über die interne Loopback Adresse des Betriebssystems verwendet. Dazu werden ein Twincat Modbus-Server und ein WinCC OA Modbus-Client benutzt. Die SPS-Funktionalität bildet eine Soft-SPS [16] ab. Diese kann mithilfe des Twincat Automation-Interface ferngesteuert werden [2].

[16] Eine Soft-SPS besteht aus einer SPS-Software, welche in der Lage ist Programme für eine Speicherprogrammierbare-Steuerung auf einem PC ablaufen zu lassen. Dabei läuft die SPS als Programm auf dem Betriebssystem des PCs. Oft werden Echtzeiteigenschaften durch spezielle Methoden hinzugefügt. [11]

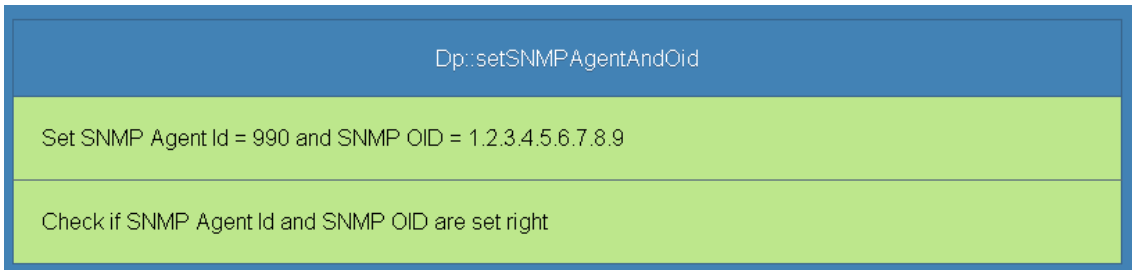

Abbildung 6.3: Test als Vergleich mit dem Erwartungswert

6.2.1 SPS-Testsystem

Das Twincat Automation-Interface ermöglicht die automatisierte Konfiguration des Twincat-Systems über Scripts. Somit ist es möglich die Tests automatisch am Test-Server in die Soft-SPS zu laden und auszuführen. Außerdem kann die Hardware-Konfiguration automatisch parametriert werden, sodass auch Hardware-Komponenten auf der Soft-SPS simuliert und getestet werden können. Um die Handhabung der Tests auf der SPS-Ebene zu vereinfachen wurde eine Software-Bibliothek mit diversen Funktionen zur SPS- und System-Konfiguration erstellt. Die Bibliothek enthält außerdem Funktionen zum Laden und Starten der Programme auf der SPS. Da die Abfrage von Zuständen innerhalb der SPS über das Automation Interface nicht möglich ist, wurde diese Aufgabe über ein Diagnosesystem auf Modbus-TCP Basis realisiert. Dabei werden die internen Variablen über Modbus-TCP aus der Steuerung in das Leitsystem eingelesen und dort verarbeitet. So kann das Testframework des Leitsystems auch für die SPS-Tests verwendet werden. Dies erleichtert die Tests ungemein und sorgt für durchgängige Überprüfung der Codes, von der Steuerung bis ins Leitsystem [2].

6.2.2 Testframework

Das Testframework im Leitsystem wurde auf Basis der Scriptsprache Control in WinCC OA implementiert. Das Framework besteht aus mehreren Bibliotheken mit diversen Funktionen für die Systemverwaltung und zur Prüfung von Erwartungswerten. Somit ermöglicht das Test-Framework die einfache Kommunikation mit der Datenbank als auch mit den diversen Treibern zur Kommunikation mit Fremdsystemen.

Ein Test besteht aus verschiedenen Funktionsaufrufen von Funktionalitäten aus der Bibliothek. Diese Funktionen stellen entweder einen Rückgabewert bereit oder verändern Zustände im System. Diese Änderungen werden mit einem Erwartungswert verglichen. Je nachdem ob der Erwartungswert erfüllt ist oder nicht, gelingt der Test oder schlägt fehl. So wird aus einer Vielzahl von Testfällen ein durchgängiger Modultest. Ein Ausschnitt aus einem Testfall ist in Abbildung 6.3 dargestellt. Die Ergebnisse der automatischen Tests werden vom Testserver auf einer Webseite dargestellt oder können per Mail an den jeweiligen Entwickler geschickt werden. Zusätzlich können die Tests im

Leitsystem jederzeit manuell aufgerufen werden.

Derzeit arbeiten wir an einer Erweiterung der Tests mittels Docker-Containern[17]. Dabei soll das Testsystem mit einem Continuous Integration [18] Werkzeug gekoppelt werden um Test unabhängig von einem stationären Testserver zu machen. Außerdem wird die Handhabung bei Merge Requests erheblich vereinfacht.

6.2.3 Kommunikationstests

Kommunikationstests sind eine Mischung aus SPS- und Leitsystemtests. Dadurch, dass alle Testfälle auf der SPS über das Leitsystem erfasst und ausgewertet werden, wird die Kommunikation meist automatisch mit getestet. Die Kommunikation zwischen SPS und Leitsystem wird über das standardisierte Kommunikationsprotokoll Modbus-TCP abgewickelt. Dieses Protokoll ist sowohl im Beckhoff-System als auch in WinCC OA vom Hersteller bereits integriert. Die vom Hersteller bereitgestellten Funktionen werden nicht getestet. Allerdings werden Kommunikationsausfälle oder sonstige Störungen des Datenaustausches im Leitsystem und in der Steuerung verarbeitet. Diese Reaktionen werden auch in Tests entsprechend geprüft, da die korrekte Erkennung und Behandlung von Kommunikationsproblemen essentiell sind.

6.3 Fehlerbehandlung

Unter Fehlerbehandlung ist in diesem Zusammenhang die Reaktion auf Fehler in der Steuerung und im Leitsystem gemeint. Im weitesten Sinn geht es auch um Möglichkeiten zur Diagnose, Fehlervermeidung und Fehlererkennung. Da die Werkzeuge des Steuerungs-Herstellers für die Diagnose und Analyse des Laufzeitsystems nur bedingt geeignet sind, müssen für den Entwicklungsprozess aber auch für den laufenden Betrieb eigene Methoden geschaffen werden.

[17] *„Docker ist eine Open-Source-Software, die bei linuxoiden Betriebssystemen dazu verwendet werden kann, Anwendungen mithilfe von Betriebssystemvirtualisierung in Containern zu isolieren. Dies vereinfacht einerseits die Bereitstellung von Anwendungen, weil sich Container, die alle nötigen Pakete enthalten, leicht als Dateien transportieren und installieren lassen. Andererseits gewährleisten Container die Trennung der auf einem Rechner genutzten Ressourcen, sodass ein Container keinen Zugriff auf Ressourcen anderer Container hat.“* [16]

[18] *„Kontinuierliche Integration (auch fortlaufende oder permanente Integration; englisch continuous integration) ist ein Begriff aus der Software-Entwicklung, der den Prozess des fortlaufenden Zusammenfügens von Komponenten zu einer Anwendung beschreibt. Das Ziel der kontinuierlichen Integration ist die Steigerung der Softwarequalität. Typische Aktionen sind das Übersetzen und Linken der Anwendungsteile, prinzipiell können aber auch beliebige andere Operationen zur Erzeugung abgeleiteter Informationen durchgeführt werden. Üblicherweise wird dafür nicht nur das Gesamtsystem neu gebaut, sondern es werden auch automatisierte Tests durchgeführt und Softwaremetriken erstellt. Der gesamte Vorgang wird automatisch ausgelöst durch Einchecken in die Versionsverwaltung. Eine vereinfachte Variante der kontinuierlichen Integration – und häufig ihre Vorstufe – ist der Nightly Build (nächtlicher Erstellungsprozess). In der Praxis trifft man auch auf kontinuierliche Integration, gepaart mit einem Nightly Build, um die Vorteile beider Systeme zu kombinieren.“* [19]

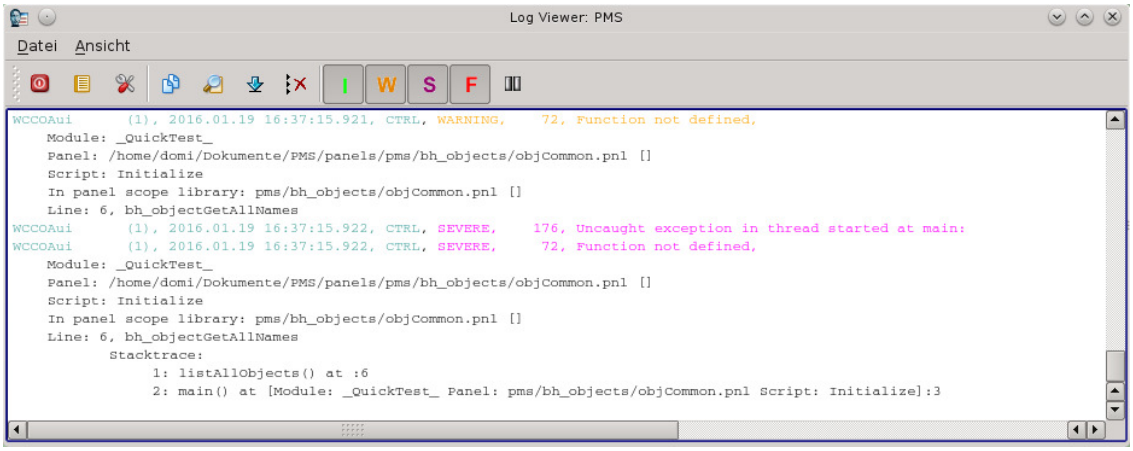

Abbildung 6.4: Viewer zur Anzeige der aufbereiteten Fehlercodes

Die wichtigste Forderung an das Laufzeitsystem ist, dass die Software-Objekte in der Steuerung auch bei einem Ausfall der Kommunikation zwischen Leitsystem und Steuerung weiter arbeiten. Dabei ist das konkrete Verhalten natürlich vom jeweiligen Objekt abhängig. Je nachdem ob und wie das Objekt mit dem Leitsystem interagiert, wird ein definiertes Verhalten erwartet. Das Verhalten wird dabei durch den Laufzeitcode des Objektes selbst bestimmt. Es muss jedoch sichergestellt sein, dass der Code unabhängig vom Zustand der Steuerung korrekt abgearbeitet wird. Solange das Steuerungssystem selbst korrekt funktioniert muss also auch das Laufzeitsystem ein deterministisches Verhalten zeigen. Dieses Verhalten wird insbesondere durch die ausgiebigen Tests, wie in Abschnitt 6 beschrieben, sichergestellt. Grundsätzlich versucht das System Fehler selbstständig zu korrigieren bzw. die Ausführung des Laufzeitcodes stets fortzusetzen. Um auch in Laufzeitcodes auf eventuelle Fehler zu reagieren, wurden vier Fehlerklassen definiert. Diese sind an die Fehlerklassen des Leitsystems angelehnt.

- INFO: Diese Klasse stellt eine Meldung dar. Die Ausführung des Systems kann unbehelligt weiter laufen.
- WARNING: Eine Warnung informiert über einen abnormalen Zustand. Es findet keine Beeinflussung der Programmausführung statt.
- SEVERE: Hierbei handelt es sich um einen ernsten Fehler. Teilfunktionalitäten können nicht bearbeitet werden. Das System versucht die Ausführung des Programmes trotzdem weiterzuführen.
- FATAL: Ein fataler Fehler kann nicht korrigiert werden und führt unweigerlich zum Abbruch der Bearbeitung des betroffenen Objektes.

Diese Fehlerklassen werden vom Laufzeitsystem über Flags zur Verfügung gestellt. Somit hat der Entwickler eines Objektes die Möglichkeit in seinem Programm auf Fehler zu reagieren. Je nach Schwere des Fehlers kann das Laufzeitsystem das Objekt weiter ausführen oder stoppt dessen Bearbeitung. Zusätzlich zu den Flags werden weiterführende Informationen zum Fehler über definierte Fehlercodes vom Laufzeitsystem über-

geben. Die Fehlercodes ermöglichen die Reaktion auf bestimmte Fehler. Zusätzlich sind die Fehlercodes für die Diagnose bei Problemen oder bei der Entwicklung von Objekten für das Laufzeitsystem sehr hilfreich.

Zur einfachen Diagnose werden die Fehler-Flags und die Fehlercodes auch über die Kommunikationsschicht ins Leitsystem übertragen. Dazu stellt das Laufzeitsystem alle Fehler in einen Fehlerpuffer. Dabei handelt es sich um einen Ringpuffer mit definierter Größe. Das bedeutet, dass die Fehlermeldungen vom System immer wieder überschrieben werden. Dieser Puffer wird vom Leitsystem ausgelesen und über die Standard-Logging-Werkzeuge von WinCC OA zur Verfügung gestellt. Durch die Einbindung der Diagnose des Laufzeitsystems in die Diagnose des Leitsystems wird eine einfache und durchgängige Fehlerkontrolle ermöglicht.

6.4 Systemumstellung

Für eine gelungene Systemeinführung ist vor allem die rechtzeitige Einbindung aller Beteiligten besonders wichtig. Deshalb wurde bei diesem Projekt versucht, die betroffenen Nutzer möglichst frühzeitig zu beteiligen. Schon bei der Zieldefinition wurden einige Nutzer einbezogen. Hierbei stellte sich sehr schnell heraus, dass das Vorhaben für viele Kollegen schwer zu überblicken war. Viele konnten sich unter den definierten Funktionalitäten nur wenig vorstellen. Um diesem Umstand entgegen zu wirken, wurde bereits zum Projektbeginn eine Testumgebung mit Steuerungs- und Systemkomponenten eingerichtet. Dadurch konnte für die Umsetzung ein evolutionärer Entwicklungsansatz gewählt werden. Durch das Testsystem war die Entwicklung aller Komponenten ohne Beeinflussung des Produktivsystems möglich. Außerdem konnten alle Entwicklungsschritte umgehend von den Entwicklern und den Anwendern getestet werden. Das Testsystem ist im Abschnitt 6 genauer beschrieben.

Nach Abschluss der wichtigsten Meilensteine wurden Schulungen für die betroffenen Nutzergruppen durchgeführt. Auch hier zeigten sich einige Vorteile durch das Testsystem. Die Benutzer konnten alle Funktionen in Ruhe ausprobieren. Dieser Umstand nimmt vor allem die Furcht vor Fehlern. Im Gegensatz zur Einführung der Verteilten-Feldebene (siehe Praxisprojekt „Verteilte Feldebene für das Plansee-Meldesystem“ [23]) konnte die Einführung auf Seiten der Benutzer schnell und reibungslos von statten gehen.

Während der Weihnachtsbetriebsruhe wurde die Systemumstellung mit einer zweiwöchigen Probephase durchgeführt. Dabei wurden alle neuen Module im Leitsystem bereitgestellt und getestet. Dadurch, dass das Laufzeitsystem von den bestehenden Modulen nahezu unabhängig ist, verlief die Umstellung nahezu reibungslos. Alle Änderungen der Funktionalitäten erfolgen im Hintergrund und sind für die Bediener in den Leitwarten transparent.

6.5 Umstellung der Steuerungen

Da das Laufzeitsystem nur als Gesamtsystem in Verbindung mit dem Leitsystem und den Steuerungen funktioniert, müssen alle Systemkomponenten entsprechend angepasst werden. Durch das Laufzeitsystem wird die lokale Funktionalität auf allen Steuerung vollständig verändert. Die direkte Kommunikation vom Leitsystem mit den Ein- und Ausgangsbaugruppen ist von der Umstellung aber nicht betroffen.

Allerdings sind einige komplexe Funktionalitäten, wie die Kommunikation zu den Brandmeldeanlagen, direkt auf den Steuerungen implementiert. Diese Funktionalitäten sind in herkömmlicher Weise, ohne Laufzeitsystem, programmiert worden. Es ist nicht Ziel der Umstellung, all diese bewährten Funktionalitäten durch das Laufzeitsystem zu ersetzen. In Zukunft sollen aber alle lokalen Funktionalitäten der Steuerungen über das Laufzeitsystem abgewickelt werden. Im gegebenen Projektzeitraum war dies aber nicht vollständig möglich. Um den notwendigen Parallelbetrieb von alten und neuen Funktionalitäten zu gewährleisten wurden die bestehenden herkömmlichen Funktionen einfach im Programm zu den Instanzen des Laufzeitsystems hinzugefügt. So bleiben die gewohnten Funktionen eins zu eins erhalten und es können neue Funktionalitäten hinzugefügt werden.

Durch den gemeinsamen Betrieb von bestehenden Funktionalitäten ohne Laufzeitsystem und den neuen Funktionalitäten des Laufzeitsystems wurde die Inbetriebnahme auf den Steuerungen immens vereinfacht. Das Laufzeitsystem konnte, wie in den Tests zuvor, ohne Probleme auf alle Steuerungen ausgerollt werden. Jene bestehenden Funktionen, welche bereits durch das Laufzeitsystem abgedeckt werden konnten, wurden umgestellt. Alle anderen Funktionalitäten wurden belassen. Diese Konfiguration läuft seit der Umstellung stabil und erfüllt die Erwartungen.

6.6 Darstellung einer konkreten Implementierung

Als konkrete Implementierung soll die Funktionalität der Ansteuerung eines Notstromaggregates gezeigt werden. Die Randbedingungen der Implementierung sind in Abschnitt 3.4 beschrieben. Ein solches Aggregat verfügt über folgende Elemente:

- Handbetrieb - Eingang
- Störung - Eingang
- Betriebsmeldung - Eingang
- Anforderung - Ausgang
- Sollwertvorgabe - Ausgang
- Fernstart über das Leitsystem - internen Zustand
- Fernstart zur Spitzenstromoptimierung - interner Zustand
- Gesamtbetriebsstunden - interner Zustand

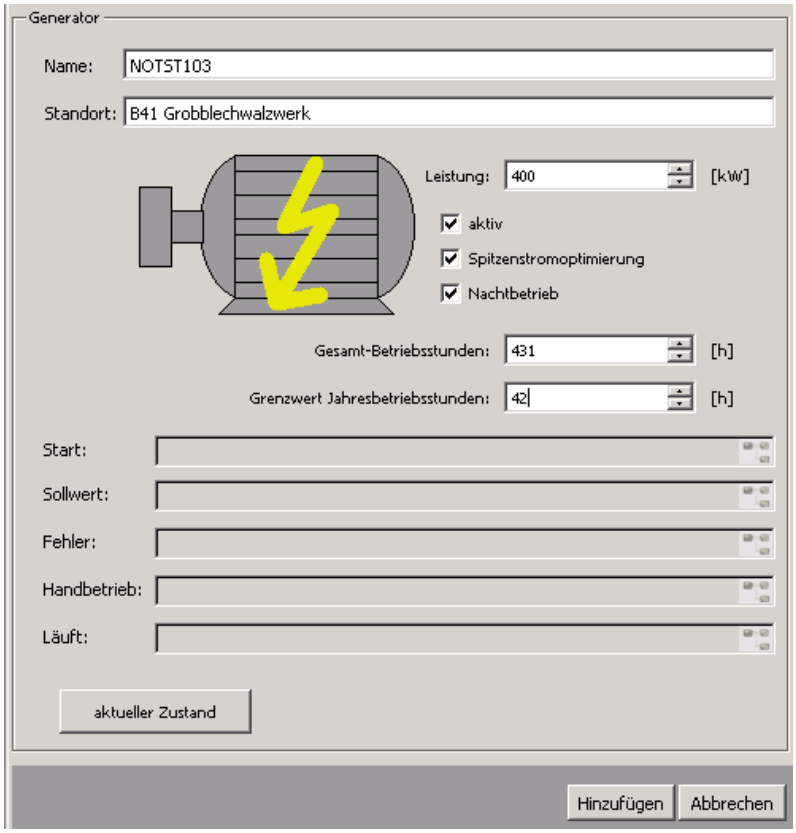

Abbildung 6.5: Parametrierung eines Notstromaggregates mit Verknüpfung der Ein- und Ausgänge

- Jahresbetriebsstunden - interner Zustand

Die gewünschte Funktionalität des Notstromaggregates kann im Laufzeitsystem sehr einfach abgebildet werden. Alle Betriebszustände werden direkt an das Leitsystem durchgereicht. Zur Ansteuerung des Aggregates wird eine einfache Rampe mit einer definierten Steigung berechnet. Für die Erfassung der Betriebsstunden wird die Zeit aufsummiert, während der das Aggregat in Betrieb ist. Auch die internen Werte des Aggregat-Objektes werden über den Objektdatenbereich an das Leitsystem übertragen. Steuerbefehle schreibt das Leitsystem wiederum in den Objektdatenbereich des betreffenden Notstrom-Objektes auf der Steuerung.

Für den Benutzer sind das Anlegen und die Konfiguration eines Notstromaggregates denkbar einfach. Über die Eingabefelder werden die benötigten Informationen zum Aggregat eingegeben und die Verknüpfungen zu den Ein- und Ausgängen parametriert. Der Parametrierdialog für die Notstromaggregate wird in Abbildung 6.5 dargestellt. Anschließend kann der Benutzer die Einstellungen über einen weiteren Dialog auf die Steuerung schreiben und das Aggregat testen. Alle Einstellungen sowohl im Leitsystem als auch auf der Steuerung werden automatisch konfiguriert.

7 Zusammenfassung der Ergebnisse

Obwohl die Realisierung des Laufzeitsystems in der vorliegenden Form noch nicht die optimale Lösung darstellt, konnten doch die Funktion und die Sinnhaftigkeit gezeigt werden. Die Systemtechniker haben nun bisher ungeahnte Möglichkeiten um Funktionalitäten auf den Steuerungen einfach zu implementieren. Es ist weniger spezialisiertes Know-How über verschiedenste Systeme notwendig, was vor allem den Bereitschaftsdienst erheblich vereinfacht. Das Instandhaltungspersonal benötigt keinen direkten Zugang auf das Steuerungsprogramm mehr. Bisher mussten sich die Mitarbeiter dazu in mehrere Systeme einarbeiten. Durch die klar vorgegebenen Funktionalitäten werden Bedienungsfehler minimiert und die Sicherheit maximiert. Durch die erheblich einfachere Diagnose von Fehlern können außerdem Stillstandszeiten von wichtigen Systemen und Anlagen verkürzt werden.

Hauptsächlich können durch die vorliegende Arbeit die Vorteile von standardisierten Prozessen und Vorgehensweisen in der Software und Systementwicklung gezeigt werden. Durch die konsequente Vereinheitlichung und Weiterentwicklung der Systeme wird die Komplexität maßgeblich reduziert. Die einfache und sichere Handhabung von Steuerungssystemen bietet einem großen Nutzerkreis mehr Möglichkeiten selbständig zu arbeiten und Probleme zu lösen. Es konnte auch gezeigt werden, dass Methoden aus der Softwareentwicklung auch für industrielle Anlagen sinnvoll und brauchbar sind. Bisher werden moderne Entwicklungswerkzeuge und automatisierte Tests in der Steuerungsentwicklung nur selten eingesetzt. Viele Werkzeuge in diesem Bereich bieten außerdem ungenügende Unterstützung für diese neuen Arbeitsmethoden.

7.1 Eigenentwicklungen

Die Laufzeitsysteme für die Beckhoff-Steuerungen und für das Leitsystem WinCC OA wurden von mir entwickelt und umgesetzt. Beim Entwurf habe ich versucht, mich an bekannten Systemen, im speziellen Forth, zu orientieren und gängige Konzepte zu übernehmen. Durch spezielle Randbedingungen mussten aber doch einige individuelle Anpassungen für die Stack-Verarbeitung vorgenommen werden. Im Zuge der Arbeit wurden auch die notwendigen Bedienoberflächen für die Interaktion mit dem Laufzeitsystem der Steuerungen erstellt. Es wurden einige kleinere Objekte für das Laufzeitsystem erstellt, welche in verschiedenen Anwendungen zum Einsatz kommen. Als konkrete und abgeschlossene Implementierung wurde in dieser Arbeit die Steuerung der Notstromaggregate in Abschnitt 3.4 dargestellt. Dazu wurden auch weitere Leitsystem-Module entwickelt, welche jedoch nicht Teil dieser Arbeit sind.

Im Bereich der Systementwicklung wurden einige Methoden von externen Firmen über-

nommen und an die Bedürfnisse von Plansee angepasst. Die strukturierte Vorgehensweise bei der Softwareentwicklung wurde bereits im vorangegangenen Praxisprojekt „Verteilte Feldebene für das Plansee-Meldesystem“ [23] begonnen. In diesem Projekt wurde diese weiterentwickelt und um einige Konzepte ergänzt. Einige dieser Konzepte sind im Abschnitt 1.3.2 beschrieben.

7.2 Innovation

Als Neuheit kann sicherlich die Herstellerunabhängigkeit der Laufzeitumgebung auf der Steuerung angeführt werden. Alle Software-Komponenten für die Steuerungen wurden auf Basis von IEC 61131 entwickelt. Auf den Steuerungen bleibt das System des jeweiligen Herstellers gänzlich unangetastet, wodurch alle Sicherheits- und Gewährleistungsvereinbarungen aufrecht bleiben. Es wurden bereits kleinere Tests mit einer Siemens S7-1500 durchgeführt, welche die Einsatzmöglichkeiten und die einfache Portierbarkeit auf andere Steuerungssysteme zeigen.

Innovativ im Kontext von Plansee ist auch die konsequente Umsetzung von Methoden aus der Softwareentwicklung für das Einsatzgebiet industrieller Steuerungssysteme. Es gibt allerdings bereits einige Arbeiten, welche ähnliche Vorgehensweisen beschreiben. Siehe dazu auch [2] und [31].

7.3 Ausblick

Langfristig stellt die Implementierung eines Forth-Systems auf einer Speicherprogrammierbaren-Steuerung eine äußerst flexible und innovative Lösung dar. Obwohl Forth heutzutage nicht mehr so stark verbreitet ist, bietet es doch viele Möglichkeiten ein flexibles und parametrierbares Steuerungssystem zu realisieren. Die Arbeitsweise von Forth bietet sich besonders für die grafische Programmierung an. Eine grafische Programmeingabe stellt dabei eine weitere Zukunftsvision in diesem Bereich dar. Durch die Anzeige einer CFC-ähnlichen Oberfläche, können Programme sehr einfach und intuitiv erstellt werden. Außerdem eignet sich diese grafisch Ansicht optimal für die Darstellung von aktuellen Zustandsinformationen. So könnte eine, aus konventionellen Werkzeugen zur Programmierung von Speicherprogrammierbaren-Steuerungen bekannte, Methode zur Erstellung von Programmen für das Forth-System entstehen.

Natürlich ist auch die weitere Verbesserung der Entwicklungsmethoden geplant. Wie schon in Kapitel 6.2.2 angesprochen, ist derzeit eine Container-basierte Lösung für die automatisierten Tests in Arbeit. Diese wird gemeinsam mit einer Partnerfirma geplant und umgesetzt. Auch das Projektmanagement und die Teamkoordination bei großen Projekten sind weiter ausbaufähig.

Literaturverzeichnis

[1] 3S-SMART SOFTWARE SOLUTIONS GMBH: *CODESYS - Das System*, Juni 2015.

[2] BATHELT, JENS: *Entwicklungsmethodik für SPS-gesteuerte mechatronische Systeme*. VAusgabe 405 von Fortschritt-Berichte VDI / 20, Verein Deutscher Ingenieure, 1997.

[3] BECKHOFF AUTOMATION GMBH: *TwinCAT 3 | eXtended Automation (XA)*. Twincat3 Flyer, 2012.

[4] BECKHOFF AUTOMATION GMBH: *New Automation Technology*, November 2014.

[5] BECKHOFF AUTOMATION GMBH: *Continuous Function Chart Editor (CFC)*, November 2015.

[6] BRODIE, LEO: *Starting FORTH*. Prentice-Hall, 1981.

[7] CERN ACCELERATING SCIENCE: *About CERN*, September 2015.

[8] CERN ACCELERATING SCIENCE: *UNICOS (UNified Industrial Control System)*, September 2015.

[9] CHACON, SCOTT: *Pro Git*. Apress, 2009.

[10] DE.WIKIPEDIA.ORG: *Qt (Bibliothek)*, November 2014.

[11] DE.WIKIPEDIA.ORG: *Soft-SPS*, Mai 2014.

[12] DE.WIKIPEDIA.ORG: *Speicherprogrammierbare Steuerung*, November 2014.

[13] DE.WIKIPEDIA.ORG: *Supervisory Control and Data Acquisition*, November 2014.

[14] DE.WIKIPEDIA.ORG: *BACnet*, Oktober 2015.

[15] DE.WIKIPEDIA.ORG: *Continuous Function Chart*, November 2015.

[16] DE.WIKIPEDIA.ORG: *Docker (Software)*, Dezember 2015.

[17] DE.WIKIPEDIA.ORG: *EN 61131*, Mai 2015.

[18] DE.WIKIPEDIA.ORG: *Forth (Programmiersprache)*, September 2015.

[19] DE.WIKIPEDIA.ORG: *Kontinuierliche Integration*, Dezember 2015.

[20] DE.WIKIPEDIA.ORG: *Parser*, Oktober 2015.

[21] DE.WIKIPEDIA.ORG: *Wiki*, November 2015.

[22] DE.WIKIPEDIA.ORG: *Umgekehrte polnische Notation*, Januar 2016.

[23] DOMINIK, BAUMANN: *Verteilte Feldeben für das Plansee-Meldesystem*. Praxisprojekt II, Hochschule Mittweida, 2014.

[24] ETM PROFESSIONAL CONTROL GMBH: *SIMATIC WinCC Open Architecture*, November 2014.

[25] EVON GMBH: *XAMControl WHITEPAPER*. XAMControl WHITEPAPER, 2014.

[26] FORTH-GESELLSCHAFT E.V.: *Was ist Forth*, September 2015.

[27] FORTH INTEREST GROUP (FIG): *What is Forth?*, März 2013.

[28] GAYET PH., BARILLÈRE R.: *UNICOS A FRAMEWORK TO BUILD INDUSTRY LIKE CONTROL SYSTEMS: PRINCIPLES & METHODOLOGY*. CERN, Geneva, Switzerland, 2009.

[29] JCOP FRAMEWORK TEAM: *JOINT CONTROLS PROJECT (JCOP) FRAMEWORK SUB-PROJECT GUIDELINES AND CONVENTIONS*. CERN, Geneva Switzerland, 2010.

[30] JCOP FRAMEWORK TEAM: *JCOP Framework Documentation*, Mai 2013.

[31] KALLENBACH E., SAFFERT E., SCHÄFFEL C. UND BIRLI O.: *Zur Gestaltung integrierter mechatronischer Produkte*. VDI Berichte 1315, 1997.

[32] LEOPOLD KLAUS, KALTENECKER SIEGFRIED: *Kanban in der IT: Eine Kultur der kontinuierlichen Verbesserung schaffen*. Carl Hanser Verlag, 2012.

[33] NXTCONTROL GMBH: *Effizientes Engineering verteilter Systeme*. Vorsprung durch Effizienz, 2014.

[34] PHILIP J. KOOPMAN, JR.: *Stack Computers: the new wave*. Ellis Horwood, 1989.

[35] PLANSEE GROUP SERVICE GMBH - GROUP COMMUNICATIONS: *Zahlen Daten Fakten 2014*. Plansee-Group: facts & figures 2014, 2014.

[36] SIEMENS AG: *SIMATIC Manual Collection*. SIMATIC Manual Collection, 2012.

Erklärung

Hiermit erkläre ich, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 29.01.2016